Script details

Run Save script

Testing API

```
async function main(workbook: ExcelScript.Workbook) {
  let iRows = 0;
  let aTemp=[]=[];

  const myHeaders = new Headers({'app-id': '##################'});
  const myInit = {
    method: 'GET',
    headers: myHeaders
  };

  let fetchResult = await fetch('https://dummyapi.io/data/v1/user?limit=10',myInit);
  let oData: dataType = await fetchResult.json();
  let json: JSONData[] = oData.data;

  json.forEach(item =>
    aTemp.push([item.id, item.title, item.firstName, item.lastName,item.picture])
  );

  iRows = json.length;
  workbook.getWorksheet('Sheet1').getRange('A2:E' + (iRows+1)).setValues(aTemp);
}

interface JSONData {
  id: number,
  firstName: string,
  lastName: string,
  title: string,
```

Notes de cours pour Microsoft Excel Office Script

V0.4 Révision 24 (2024-05-04)

Vincent Isoz

1 Table des matières

2 Bibliographie 4
3 Médiagraphie 5
4 Votre avis nous intéresse! 6
5 Introduction 7
5.1 Objectifs 8
5.2 Qu'est-ce que Office Script 9
5.3 Avantages et Désavantages 10
5.4 Office Scripts vs Office.js 13
5.5 Office Scripts vs Power BI 17
6 Rappels sur ce qu'est un langage de programmation 18
7 Activer les Office Scripts 21
7.1 Pour Excel Online 21
7.2 Pour Excel Desktop 24
8 Enregistrement de Scripts (ie macros) 26
8.1 Première macro de type formatage et saisie seuls (sans bugs/sans pièges) 29
8.1.1 Exécuter le script 30
8.1.2 Éditer le script 31
8.1.3 Aide 45
8.1.4 Options d'affichage 47
8.1.5 Décrire le script 47
8.1.6 Renommer le script 48
8.1.7 Rendre la macro accessible 50
8.1.8 Sauvegarder le script (et le rendre éditable par d'autres) 54
8.2 Macro relative et absolue 59
9 Écriture de scripts 60
9.1 Commentaires 60
9.2 Variables et constantes 65
9.2.1 Variables locales & globales 68
9.2.2 Spécificités des dates 69
9.2.3 Spécificités des Tableaux (Arrays) 69
9.2.4 Créer des dictionnaires/hashes 74
9.3 Structures algorithmiques 75
9.3.1 Structure Conditionnelle 75
9.3.2 Boucles 80
9.3.3 Fonctions récursive (passage de paramètres) 85
9.4 Fonctions mathématiques 86
9.5 Débogage 87
9.6 Gestion des erreurs 90

Public Internal

9.7 Programmation Orientée Objet 91
10 Userforms 93
11 Procédures et Fonctions Classiques 94
11.1 Modèles Office Script Microsoft 94
11.2 Lire/Écrire une valeur dans une cellule 97
11.2.1 Par sa référence 97
11.2.2 Par son index 97
11.3 Lire un nombre sous forme de texte dans un cellule et le coercer en nombre 99
11.4 Boucler dans un plage de cellules donnée 99
11.5 Boucler dans un plage de cellules donnée pour identifier des formats 100
11.6 Afficher une date correctement dans la console 100
11.7 Lister les feuilles d'un fichier 103
11.8 Renommer toutes les feuilles 104
11.9 Trier les feuilles 105
11.10 Ajouter une ligne à la fin d'une table 106
11.11 Vérifier si un fichier est en lecture seule 108
11.12 Supprimer tous les liens hypertextes d'un classeur 109
11.13 Créer une table des matières 109
11.14 Appeler une API 111
12 Performance 114
13 Édition dans Visual Studio Code 119
13.1 Visual Studio Code Desktop 119
13.2 Visual Studio Code Web 121
14 Exemples d'association avec Power Automate 124
14.1 Copie de données d'une feuille Excel à une autre 124
14.2 Copie de données d'un fichier XL à un autre et envoi par email des graphs et tables 134
14.3 Conversion de fichiers *.csv en fichiers Excel (copie de données) 165
15 Conclusion 189
16 Table des figures 190
17 Index 190

Public Internal

2 Bibliograpie

Le seul livre disponible à ce jour sur Amazon toutes langues confondues est (cela sent la moquerie d'un geek à trois kilomètres…) ce petit ouvrage de 153 pages:

Dont la version en ligne sous forme de blog est disponible ici:

https://www.samurai-emblem.com/category/article/microsoft-excel/office-scriptsmicrosoft-excel/

3 Médiagraphie

Voici quelques liens utiles:

- Documentation Office Script
  https://learn.microsoft.com/en-us/office/dev/scripts/
- Documentation Office Script pour Excel (avec lien PDF téléchargeable):
  https://learn.microsoft.com/en-us/office/dev/scripts/overview/excel
- YouTube de Sudhi Ramamurthy
  https://www.youtube.com/channel/UCPncyTU_HUNNcJS3Kol2n0w
- YouTube de Analysis Cloud IT Vlog
  https://www.youtube.com/watch?v=xWPnQhJJ8H8&list=PL2nG2TcT_3Z6pQniv9tSOdPOTbRCD8Eg4
- Forum Office Script Stackexchange
  https://stackoverflow.com/questions/tagged/office-scripts

Public Internal

4 Votre avis nous intéresse!

En tant que lecteur de ce document, vous êtes le critique et le commentateur le plus important. Votre opinion compte et il est très intéressant de savoir ce qui est bien, ce qui peut être mieux et les sujets que vous souhaiteriez voir être traités.

Vous pouvez nous envoyer un courriel pour partager ce que vous avez aimé ou détesté dans le présent document afin d'en assurer une amélioration continue.

Notez que malheureusement, nous ne pouvons répondre gratuitement à des questions techniques d'ingénierie ou de problématique d'entreprise par e-mail pour des raisons professionnelles évidentes.

E-mail: isozv@outlook.com

Public
Internal

5 Introduction

Ce support de cours est en français avec captures d'écrans en anglais car c'est ce qui correspond à la majorité du bassin de population avec lequel je travaille en Suisse.

Remarques:

- R1. Ce support est quasi-inutile pour un non initié si celui-ci n'a pas suivi le cours dispensé y relatif et qu'il n'a pas complété ce dernier par de nombreuses annotations.
- R2. Ce support constitue un "super condensé" de ce qui est vu pendant le cours et qui pourrait facilement tenir sur plusieurs milliers de pages.
- R3. Nous avons exprès introduit des erreurs et des incohérences dans le document afin d'exciter l'esprit critique des apprenants
- R5. Il peut arriver à tout moment que Microsoft décide suite à des mises à jour que certaines commandes soient désactivés. C'est ainsi et vous ne pourrez rien y faire. Il faudra réadapter votre code….

Donc:

CE DOCUMENT EST PRÉVU POUR ÊTRE ACCOMPAGNÉ D'EXPLICATIONS ORALES ET DE DOCUMENTS ANNEXES. IL EST DONC TRÈS LOIN D'ÊTRE COMPLET.

Public Internet

CECI IMPLIQUE QUE:

SI VOUS N'AVEZ PAS ASSISTÉ AU COURS, LA LECTURE DES PAGES SEULES PEUT VOUS AMENER À FAIRE DES CONTRESENS IMPORTANTS ET **DANGEREUX**

R6. Nous tenons à nous excuser du fait que ce document mélange les captures d'écran de pas mal de versions de Microsoft Excel Online et en plus de différentes langues. Effectivement, notre métier de consultant en formation professionnelle nous oblige constamment à changer d'ordinateurs et ainsi les contenus des documents que nous rédigeons. Nous espérons que le lecteur comprendra en attendant une uniformisation.

5.1 Objectifs

Microsoft Excel a connu plusieurs évolution ces 30 dernières années comme l'illustre bien la figure ci-dessous:

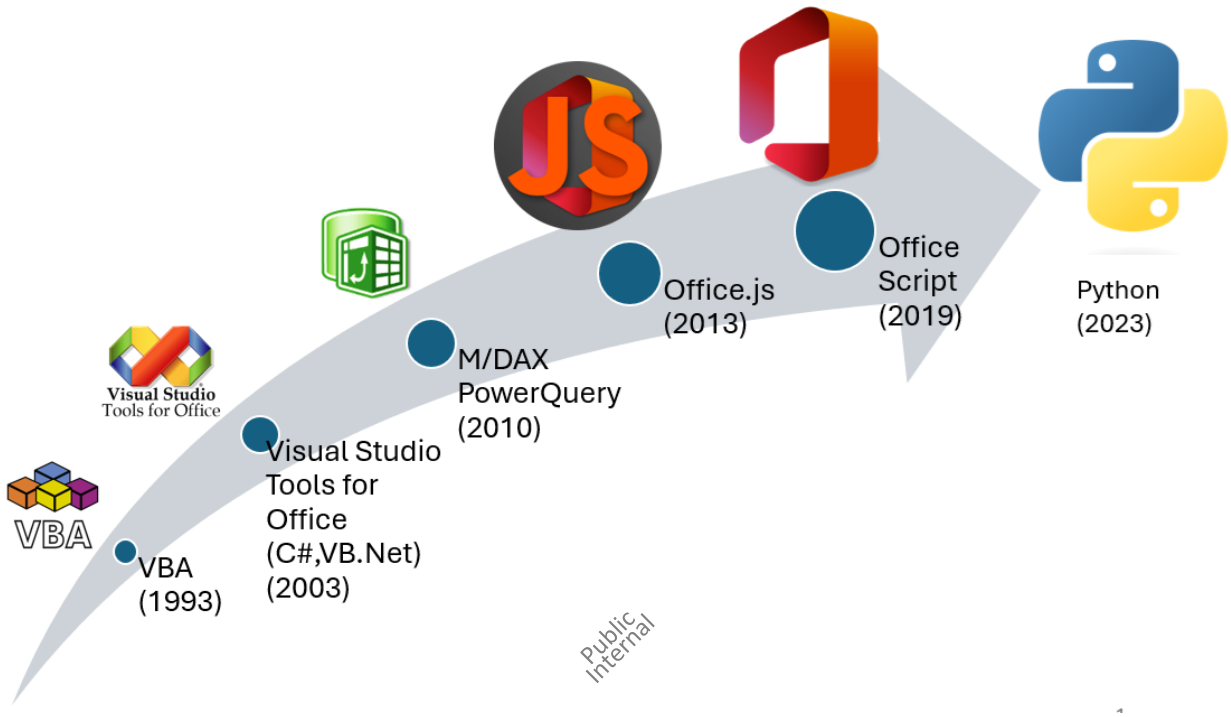


Figure 1 Évolution des langages dans Excel

A la fin de cette formation, les participants sauront très bien utiliser les macros automatiques de Office Script et connaîtront les notions et structures standards de la programmation Office Script, telles que les variables, les boucles, les conditions et les fonctions.

Ils sauront les base de ce qu'est la programmation objet et auront réalisés quelques exercices utilisant ces notions.

Ils auront travaillé avec des pseudo-formulaires et manipulé des composants utilisateurs simples comme des boutons et des champs texte.

Remarque: On sait que le nombre de mots d'une langue est limité. Le vocabulaire d'un enfant de 10 ans tourne autour de 5'000 mots, celui d'un adulte cultivé de 10'000-15'000, et les dictionnaires en plusieurs volumes peuvent monter de 130 000 à 200 000. L'Office Script d'après des estimations contiendrait environ 3'000 mots… (à vérifier quand même!).

5.2 Qu'est-ce que Office Script

Les scripts Office sont des programmes permettant d'automatiser la version web d'Excel pour ceux qui ont la **licence E3 ou E5** d'Office 365 qui ont été introduit pas Microsoft en **Janvier 2020**; en utilisant les scripts Office, vous pouvez automatiser des tâches qui seraient trop compliquées à effectuer manuellement. Par exemple, si vous effectuez la même tâche tous les jours, vous pouvez gagner du temps en l'automatisant à l'aide des scripts Office.

Les scripts Office sont utilisés pour manipuler des cellules, des feuilles et des classeurs Excel. Par exemple, ils peuvent lire ou écrire des valeurs dans des cellules Excel ou modifier la couleur d'arrière-plan des cellules. Il peut également créer, supprimer et renommer des feuilles Excel.

Les scripts Office sont basés sur un langage de programmation appelé JavaScript, qui est l'un des langages de programmation les plus faciles à apprendre pour les débutants. Comme les macros, il peut également enregistrer des opérations.

Public
Internal

5.3 Avantages et Désavantages

Comme vous le verrez dans cette formation, Office Script est beaucoup plus technique et abstrait que le VBA. Une majorité de consultants et formateurs s'accordent sur le fait que ce langage n'est pas fait pour les non-informaticiens. Ces derniers ont meilleur intérêt à aller dans Microsoft Power BI!

Avantages:

1. Fonctionne sur le web ET sur la version desktop (depuis janvier 2023)
2. Est souvent 2 à 3 fois plus rapide que le VBA
3. Permet d'ajouter des actions enregistrées

Public (Internal)

Désavantages/Problèmes:

1. Ne fonctionne qu'avec les licences E3 et E5
2. Est disponible que dans Microsoft Excel (même 3 ans après sa sortie)
3. L'éditeur de script ne permet pas la fenêtrage de codes
4. L'éditeur de script n'a pas d'explorateur d'objets, méthodes, propriétés et événements
5. Un script créé dans un fichier sera visible pour tous les fichiers Microsoft (même ceux qui ont un contenu donc qui n'a rien à voir avec l'usage de la macro)
6. Impossible d'organiser un script en modules et classes (+ un script ne peut pas appeler d'autres scripts)
7. Impossible de faire de l'automation en-dehors de ce qui est dans le cloud (il faut obligatoirement passer par Power Automate pour interagir avec les autres applications Office) et donc aussi pas de webscrapping, pas de ADODB ou OLEDB etc.

8. Impossible de faire de l'automation de Power Query, Power Pivot ou Power Map

9. ~~Impossible d'enregistrer des macros en mode relatif (seul le mode absolu existe)~~

10. Impossible de faire référence à d'autres classeurs (lire le contenu) à partir du script d'un classeur

11. Impossible de créer ses propres fonctions et catégories de fonctions

12. Impossible de faire référence au solveur ou à l'outil valeur cible ou à l'Analysis Toolpack

13. Office Script n'a pas de boîte de dialogue (message box) ni de Userform

14. ~~Impossible de créer un bouton lié à un script~~

15. Impossible de personnaliser le ruban, la barre d'accès rapide ou le menu contextuel via du code

16. Pas de procédure évènementielles (event listener)

17. Office Script ne permet de pas de rafraichîr la majorité des sources (comme PivotTable.refresh) quand utilisé via Power Automate. Seules les sources Power BI peuvent être raffraichies.

Public Internal

18. Impossible de faire du débogage avec des points d'arrêts, espions, etc.

19. Impossible de monitorer la mémoire utilisée dans utiliser les "Developer Tools" du navigateur

20. Impossible de faire du rechercher/remplacer dans un script ou dans un projet

21. Les feuilles Excel n'ont pas de "nom VBA" avec tous les problèmes que cela implique…

22. Impossible de faire appel à du C++, R ou Python externe

23. Nécessite pour tout tâche beaucoup plus d'efforts et de temps que Microsoft Power BI

24. Impossible d'envoyer un fichier Excel par e-mail à une personne en-dehors du tenant avec l'Office Script inclus dans le fichier (car les office script ne sont pas enregistrés dans les fichiers!)

25. Contrairement aux macros VBA, si l'IT veut bloquer les Office Script, ils peuvent les bloquer que pour toute l'organisation et non pas par individu…

26. Impossible de créer et d'associer de la documentation (aide) *.chm aux fonctions/routines contrairement au VBA

27. Les requêtes et réponses entre Office Script et Power Automate sont limités à 5MB

28. Une plage de traitement est limitée à 5 millions de cellules…

Public
Internal

5.4 Office Scripts vs Office.js

Il existe deux API différentes : Office.js et Office Scripts. Office.js dispose d'API disponibles sur des plateformes autres que Microsoft Excel (par exemple Word, OneNote, etc.). Les Office Script ne sont actuellement disponibles que pour Excel.

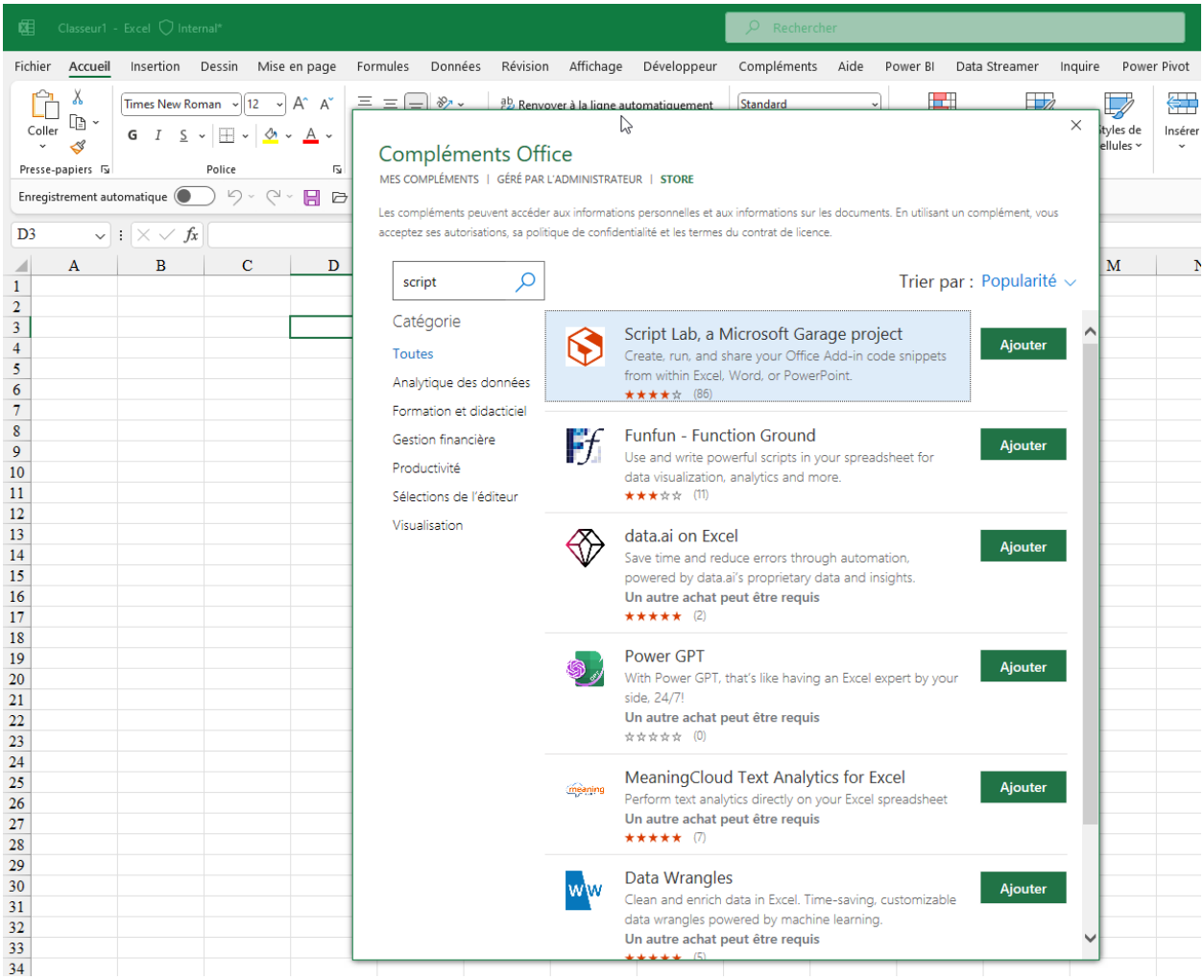
ScriptLab est un complément destiné à explorer les API Office.js. Bien que vous puissiez essayer de l'utiliser à des fins d'automatisation, il n'est pas destiné à être un environnement de développement. Vous pouvez cependant expérimenter l'API, créer des fonctions personnalisées, etc. Vous pourrez ultérieurement créer des compléments plus sophistiqués à l'aide d'un créateur de compléments Office tel que le générateur Yeoman pour Office. Les compléments seraient ensuite déployés sur un serveur où ils pourraient être utilisés sur n'importe quelle plate-forme prise en charge (PC, Mac, Web, etc.)

Public version

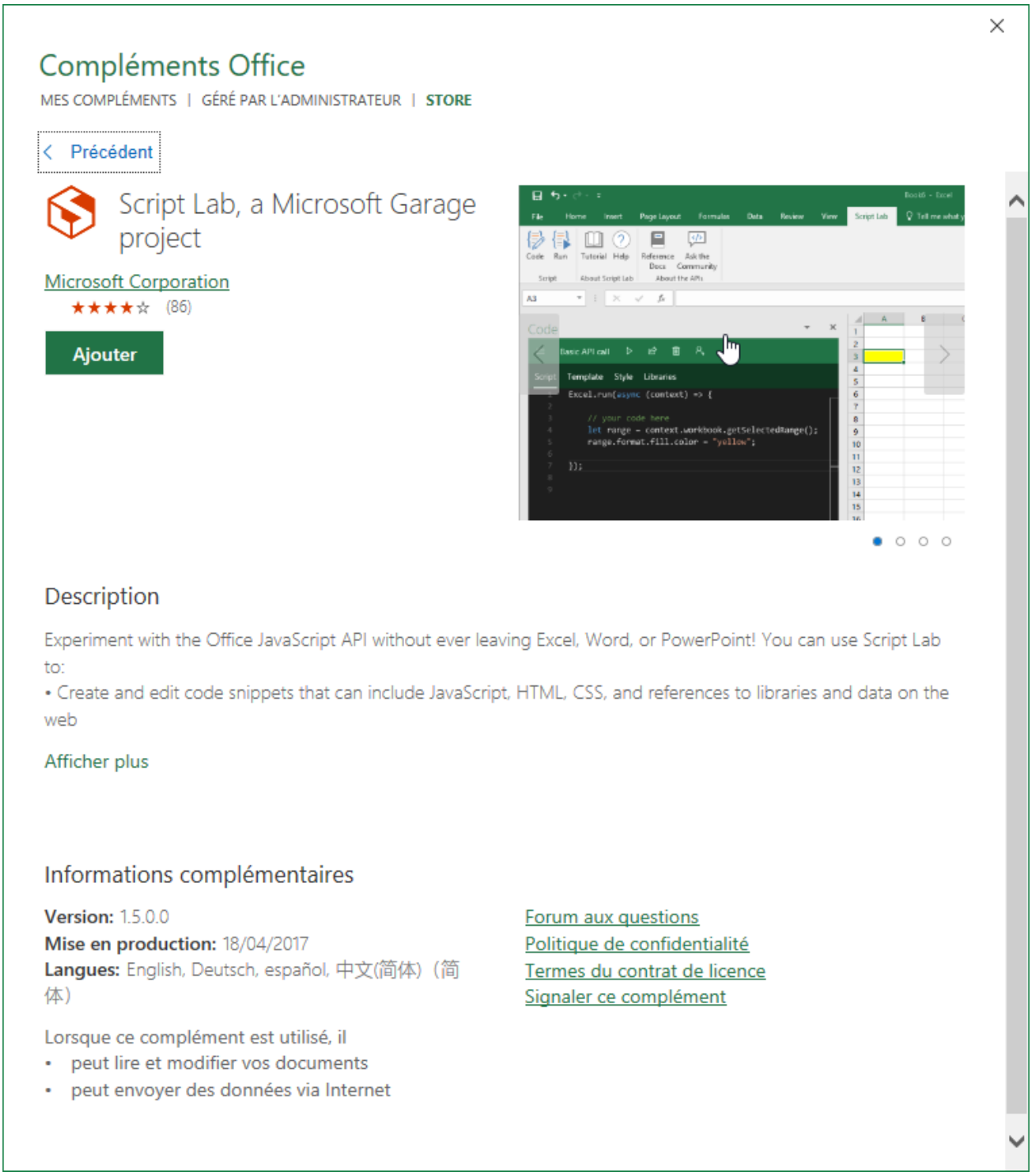
Office.js est plus adapté aux développeurs traditionnels. Pour développer des compléments, en plus de TypeScript, vous aurez peut-être également besoin de connaissances en HTML/CSS. En termes de JavaScript/TypeScript, vous devez également connaître les promesses ou les concepts asynchrones

Office Scripts est une API simplifiée. Il est en fait construit sur Office.js. Il ne nécessite pas de connaissances en HTML/CSS, promesses, concepts asynchrones, etc. Il ne nécessite pas non plus de déploiement sur un serveur web. Parce que c'est simplifié, les API finissent par être différentes. Mais dans l'ensemble, les API ne sont pas trop différentes. Si vous souhaitez effectuer une automatisation relativement simple et ne souhaitez pas gérer les frais généraux liés au développement de compléments, les scripts Office sont un bon choix. C'est également un bon choix si vous souhaitez créer des scripts simples à intégrer à PowerAutomate.

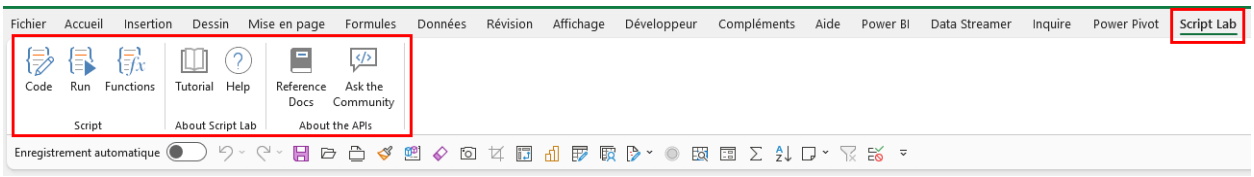
Office.js est souvent utilisé avec le complément *Script Lab* de Microsoft Excel:



Si nous cliquons sur **Script Lab**, nous avons:



Si nous cliquons sur **Ajouter**, nous avons un nouvel onglet nommé **Script Lab**:



On peut construire des scripts Office.js incluant du Python mais pas avec Office Script au jour où nous écrivons ces lignes.

Office.js gère l'évènementiel mais pas Office Script. La raison pour laquelle les événements ne fonctionnent pas est que les scripts Office sont synchrones alors que l'API javascript est asynchrone. Les API Office Scripts constituent un sous-ensemble optimisé et synchrone du modèle d'API JavaScript Excel.

Public
Internal

5.5 Office Scripts vs Power BI

Il est bizarre de vouloir utiliser Office Scripts qui nécessite le cloud et qui est extrêmement limité, quand nous avons depuis 2015 un outil comme Microsoft Power BI qui est infiniment plus flexible et versatile que Microsoft Excel???!!!

C'est probablement un problème de biais de l'outil et de biais d'ancrage qu'autre chose. Mais factuellement, voyons les raisons qui font que toute personne qui automatise des choses à intérêt à utiliser Power BI plutôt que Office Script d'Excel:

1. Microsoft Power BI (desktop et version web) gère nativement Power Query (et donc le langage M)
2. Microsoft Power BI (desktop et version web) gère nativement Power Pivot (et donc le langage DAX)
3. Microsoft Power BI (desktop et version web) gère nativement Power View (et donc le langage DAX)
4. Microsoft Power BI (desktop et version web) a un peu plus de 380 types de graphiques contre 70 basiques pour Microsoft Excel
5. Microsoft Power BI a plusieurs modules d'analyse basée sur l'intelligence artificielle contre zéro pour Office Script de Microsoft Excel
6. Microsoft Power BI desktop gère nativement TOUS les packages et libraires R et Python
7. Microsoft Power BI Service (la version web cloud) gère environ 20 libraires Python et plus de 400 packages R
8. Microsoft Power BI peut être lié à Jupiter, Spyder, PyCharm, Anaconda ou R Studio pour éditer le code.
9. Microsoft Power BI permet de faire de la RLS (détection d'utilisateur n'est pas encore disponible dans l'API Office Script)
10. Microsoft Power BI permet de se connecter à Fabric avec ses Data Lake et Modèles sémantiques
11. Microsoft Power BI intègre Power Apps ainsi que Power BI Report Builder et peut s'intégrer dans Power Pages

Public
Internal

12. Microsoft Power BI Embedded permet l'intégration des tableaux de bords et rapports dans n'importe quelle application tierce

6 Rappels sur ce qu'est un langage de programmation

Un langage de programmation est un ensemble de règles qui permettent aux humains de communiquer avec les ordinateurs. Il s'agit d'un langage formel qui utilise des mots-clés, des symboles et des règles de syntaxe pour indiquer aux ordinateurs ce qu'ils doivent faire.

Les langages de programmation peuvent être utilisés pour créer différents types de programmes informatiques, tels que des applications, des jeux, des sites Web et des logiciels. Ils sont également utilisés dans la recherche scientifique, l'ingénierie et la finance.

Il existe de nombreux langages de programmation différents, chacun avec ses propres avantages et inconvénients. Certains langages sont plus faciles à apprendre que d'autres, tandis que d'autres sont plus performants ou plus efficaces.

Voici quelques exemples de langages de programmation courants :

- Python : un langage de programmation général-usage qui est populaire pour l'apprentissage automatique, l'intelligence artificielle et le développement Web.
- Java : un langage de programmation orienté objet qui est populaire pour le développement d'applications mobiles et d'applications Web.
- C++ : un langage de programmation orienté objet qui est populaire pour le développement de logiciels de haute performance.
- C : un langage de programmation de bas niveau qui est populaire pour le développement de systèmes d'exploitation et de pilotes de périphériques.

Public
Internal

Les langages de programmation peuvent être divisés en plusieurs catégories, notamment :

- **Langages de haut niveau** : ces langages sont conçus pour être faciles à lire et à écrire pour les humains. Ils sont généralement traduits en langage machine par un compilateur ou un interpréteur.
- **Langages de bas niveau** : ces langages sont plus proches du langage machine et sont généralement plus difficiles à apprendre et à utiliser. Ils sont généralement utilisés pour le développement de logiciels de haute performance ou pour le contrôle de périphériques.

- **Langages interprétés** : ces langages sont exécutés directement par l'interpréteur sans être compilés en langage machine. Ils sont généralement plus lents que les langages compilés, mais ils sont plus faciles à déboguer.

- **Langages compilés** : ces langages sont traduits en langage machine avant d'être exécutés. Ils sont généralement plus rapides que les langages interprétés, mais ils sont plus difficiles à déboguer.

Les langages de programmation sont un outil puissant qui peut être utilisé pour créer des applications et des logiciels de toutes sortes. Ils sont essentiels au développement de la technologie moderne.

Autres points qu'il faut savoir:

- Aucun langage informatique n'est logique (ce qui est logique c'est l'électronique binaire sous-jacente)

- Un langage informatique permet d'exploiter le mécanisme déterministe d'une machine de Turing comme l'est un ordinateur classique

- Chaque langage à sa grammaire et vocabulaire propre

- Les langages informatiques, comme les langues vivantes… vivent et donc évoluent avec le temps!

Il faut donc comme à la petite école apprendre par cœur des règles et procédures qui n'ont qu'un sens apparent!

Voir https://rosettacode.org/wiki/Rosetta_Code

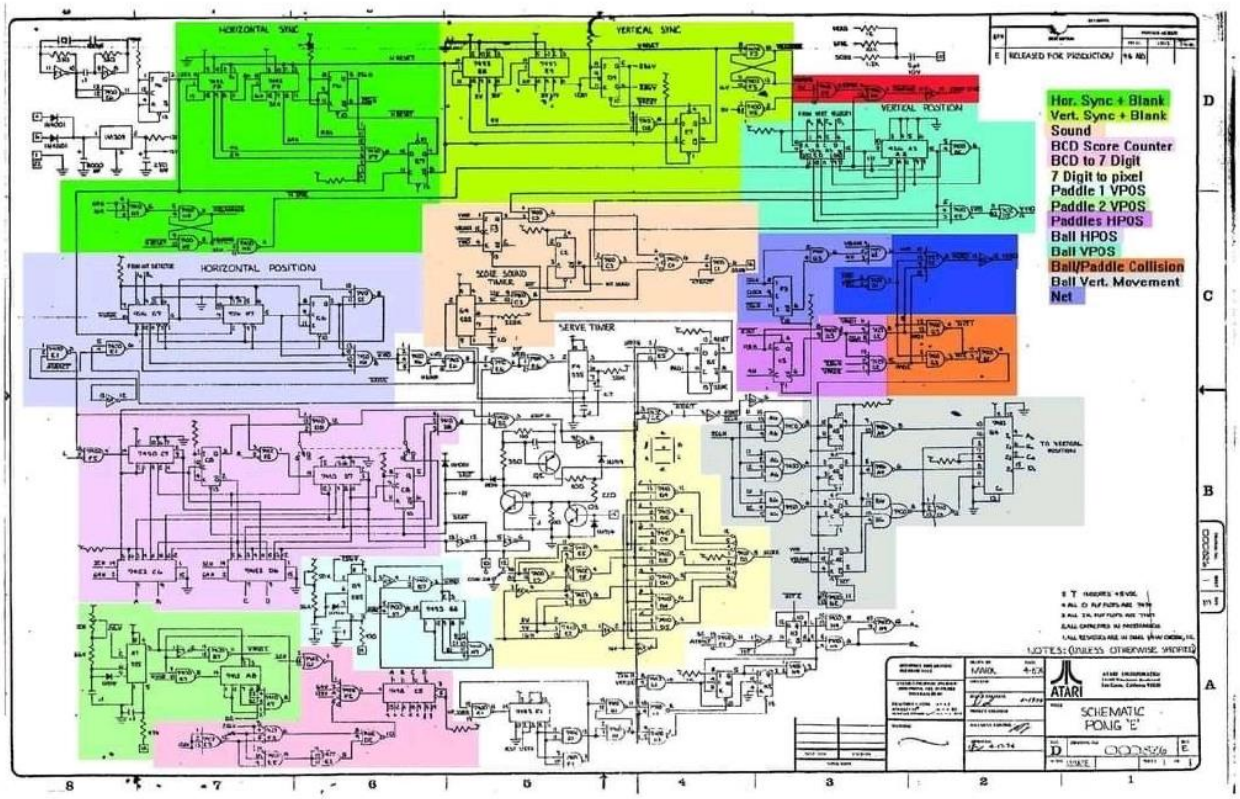


Figure 2 Pong sur ATARI en circuit logique

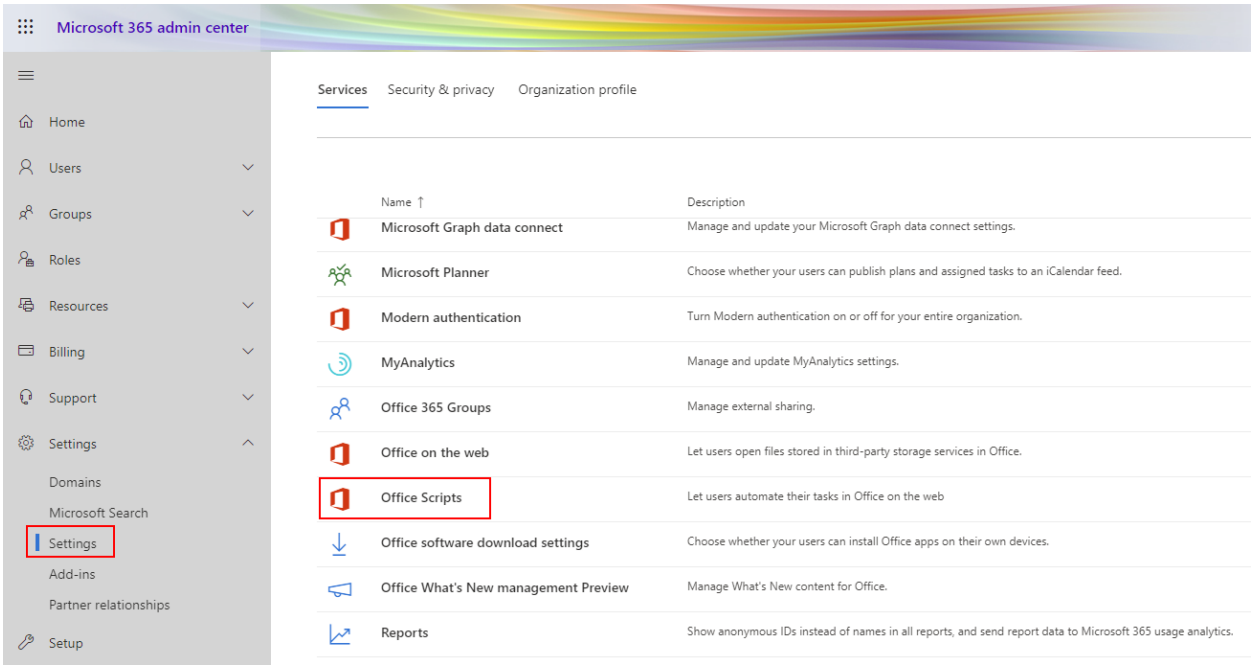
Public Internal

7 Activer les Office Scripts

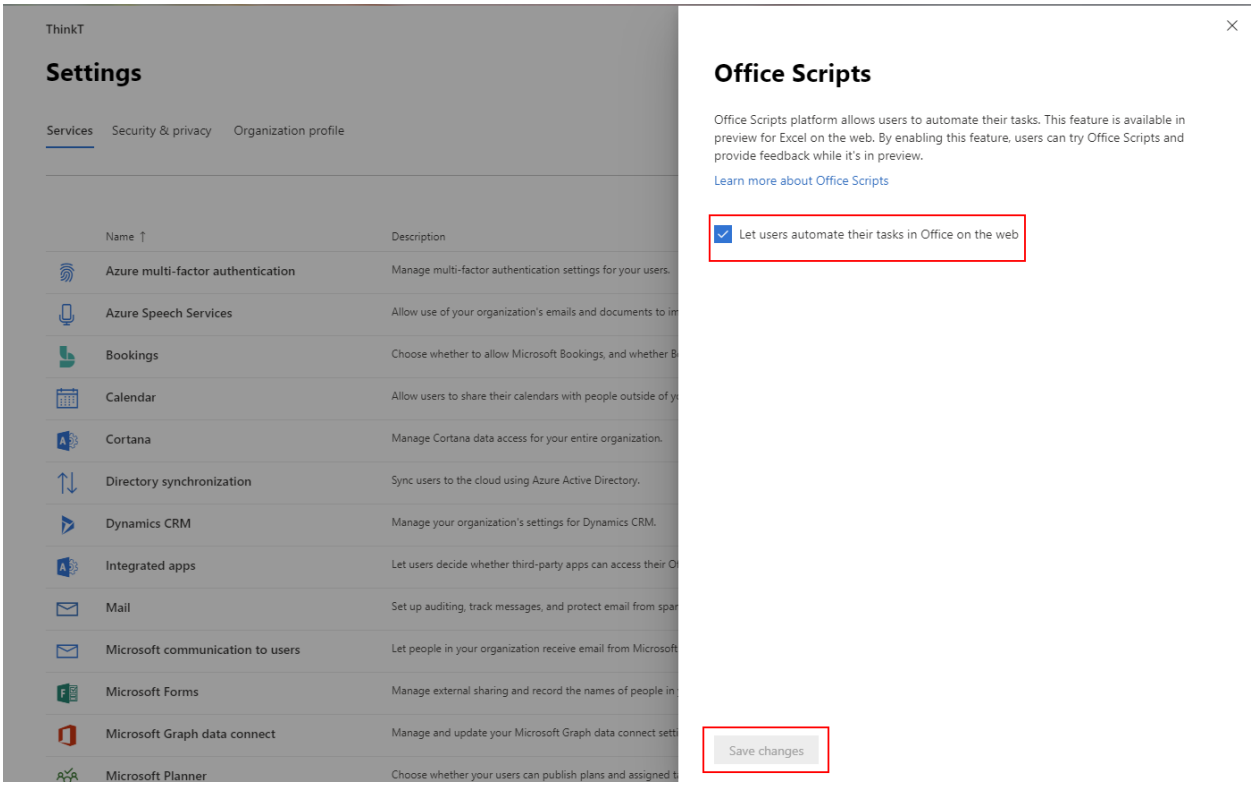
7.1 Pour Excel Online

Première chose qu'il faut savoir c'est qu'au jour où nous écrivons ces lignes (Avril 2020), les Office Scripts ne sont pas activés par défaut chez les utilisateurs.

Les administrateurs du tenant doivent aller dans l'admin center et chercher l'option **Office Scripts** visible ci-dessous:

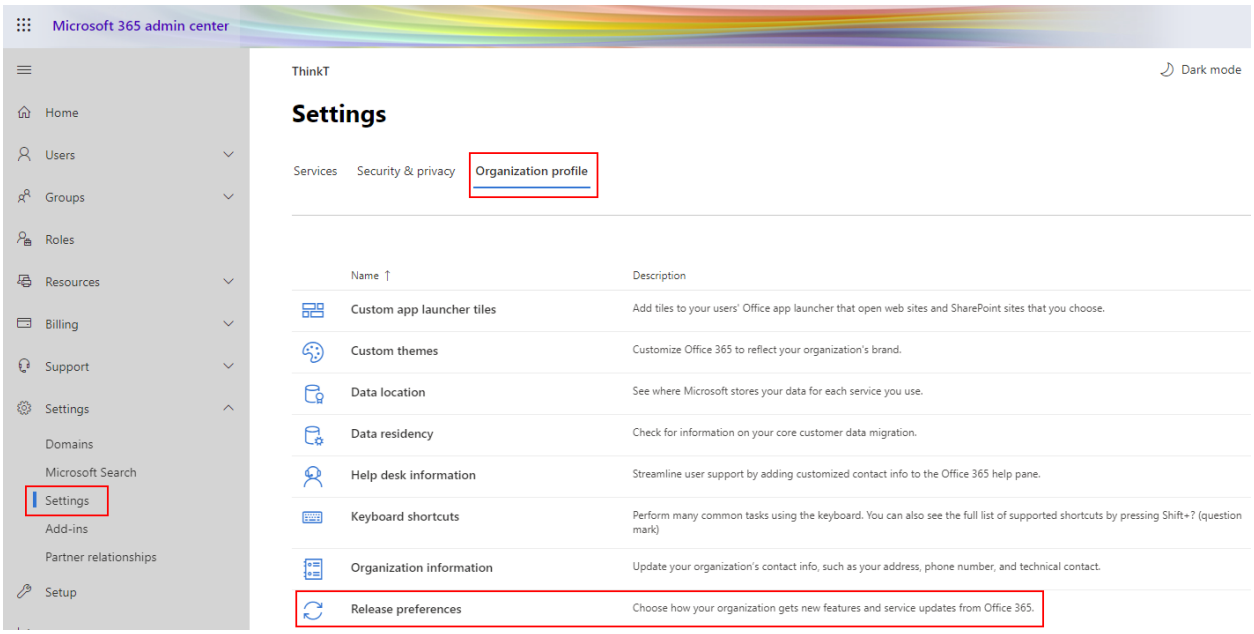


Vous cliquez sur **Let users automate tasks in Office on the web**:

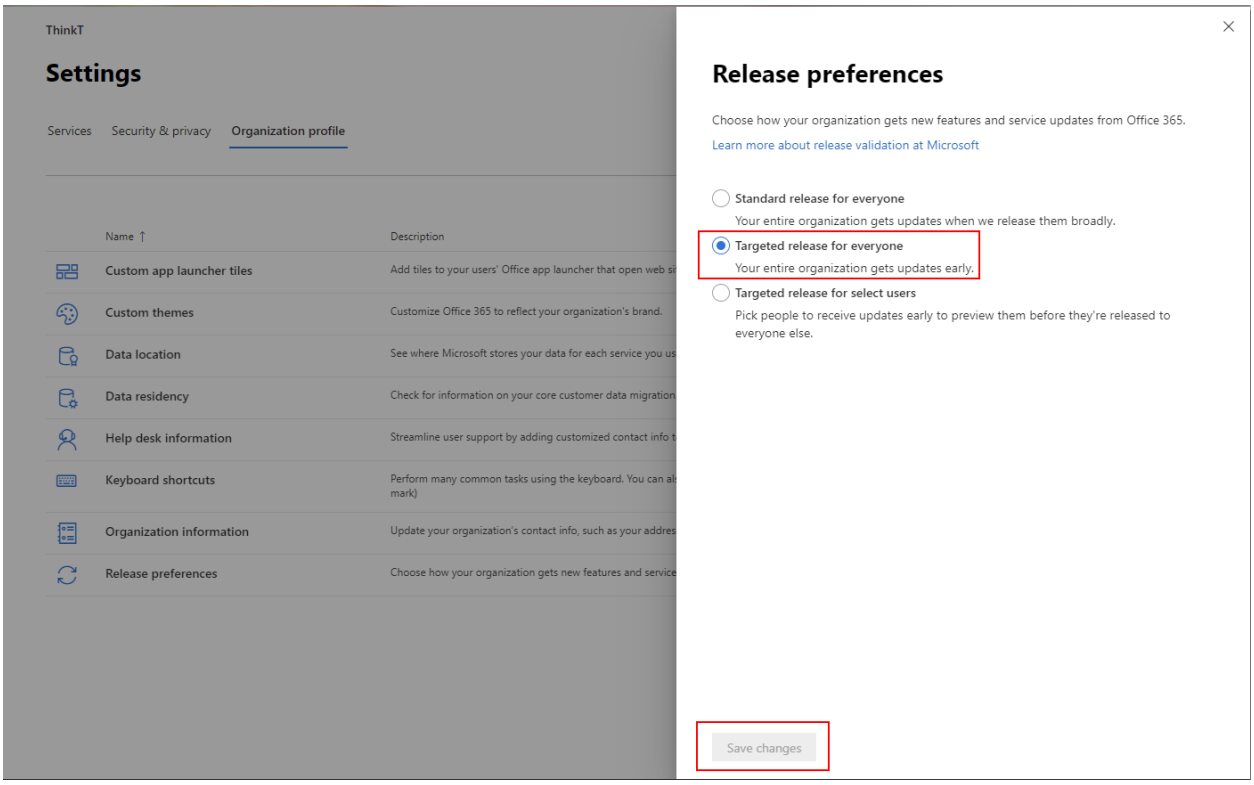


Ensuite, dans les **Settings** du tenant, vous allez dans **Settings** et cliquez sur **Release preferences**:

Public
Internal



Et vous prenez l'option **Targeted release for everyone**:



Ensuite, il faut attendre quelques minutes à au pire 48 heures basé sur notre expérience…

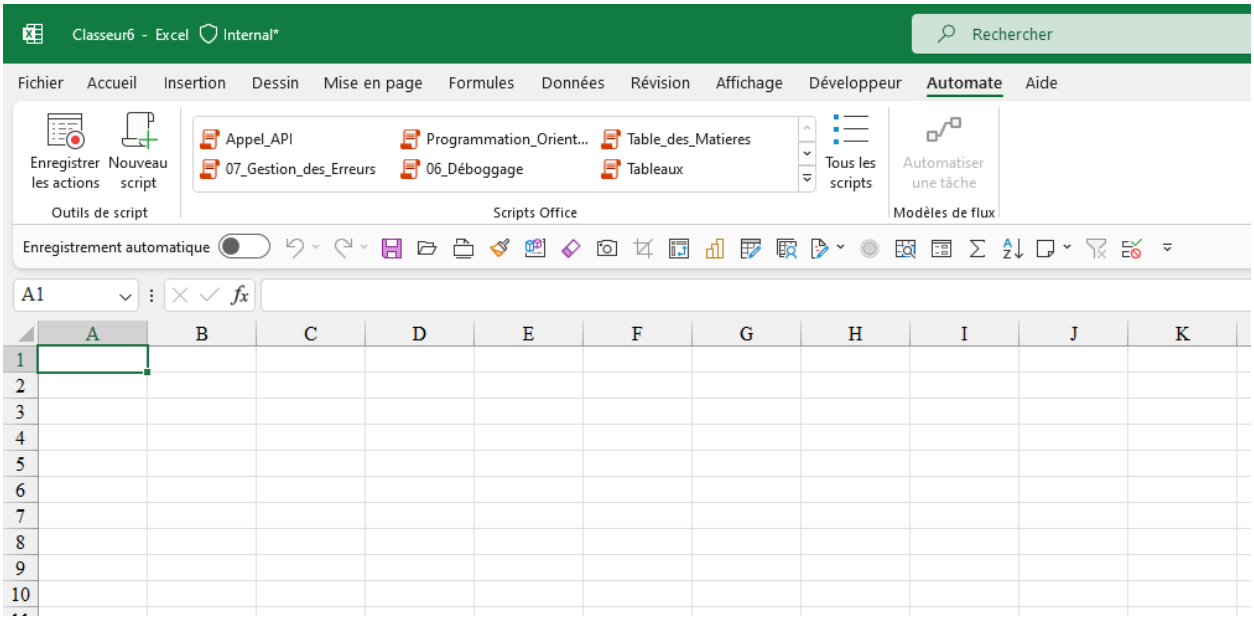
Public
Internal

7.2 Pour Excel Desktop

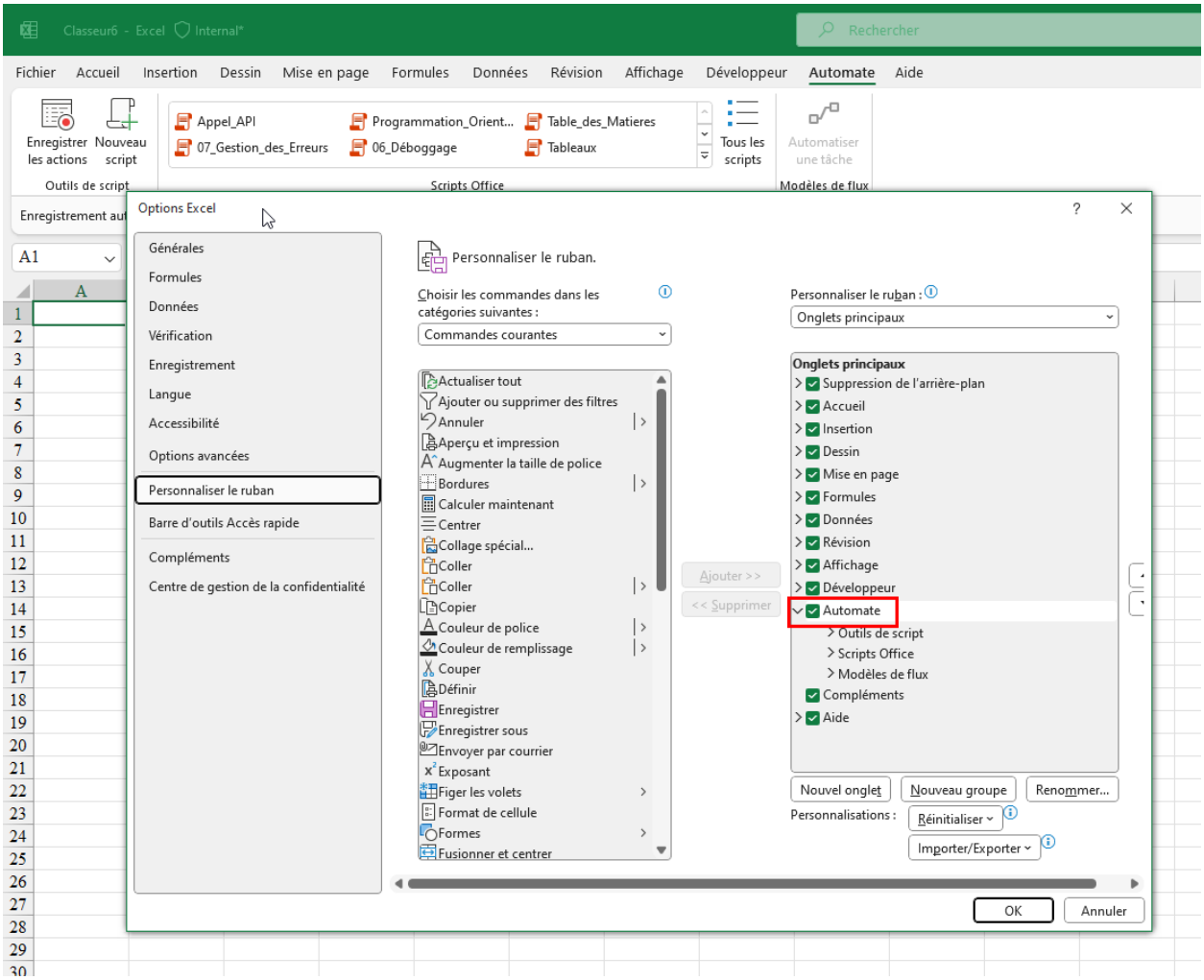
Microsoft a annoncé le **3 Janvier 2023** avoir commencé à déployer l'onglet Automate sur Microsoft Excel pour Windows et macOS. La dernière mise à jour permet aux utilisateurs d'ordinateurs de bureau de créer et de modifier des scripts Office pour automatiser leurs tâches ou flux de travail répétitifs.

Jusqu'à présent, l'onglet Automatiser n'était disponible que pour les clients dans Excel sur le Web. Avec cette version, l'application de bureau Excel aide les utilisateurs à enregistrer leurs actions sous forme de script réutilisable, puis à exécuter ce script si nécessaire pour améliorer l'efficacité.

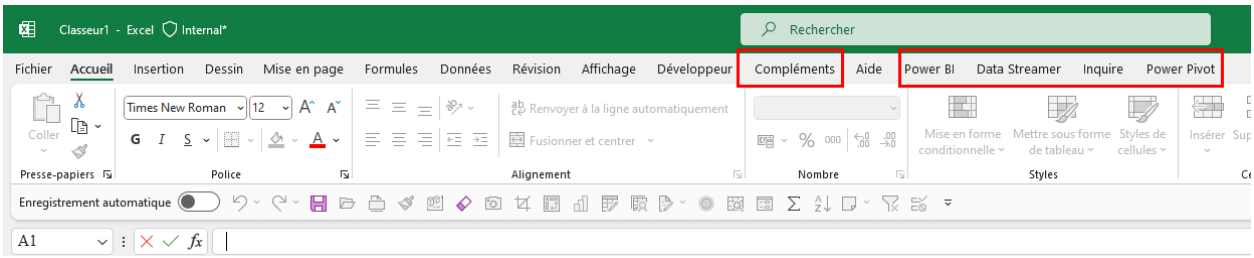
Pour l'activer il n'y a rien à faire si vous êtes sur un tenant dans un pays où il a été déjà déployé, vous aurez alors:



Si jamais il n'apparaît pas, vous pouvez vérifier toutefois s'il n'aurait pas été désactivé dans les rubans:



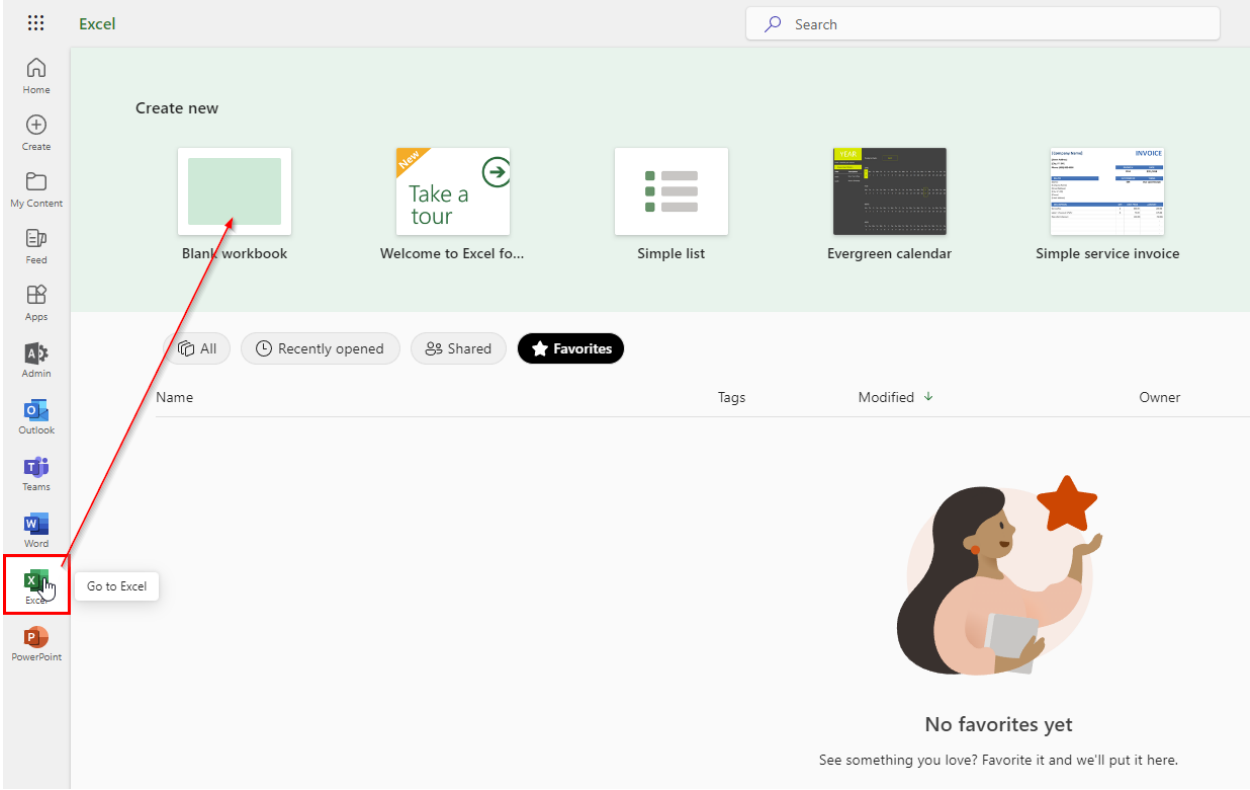
Attention il y a un bug avec cet onglet Automate. Si vous l'activez, tous les autres onglets exotiques disparaîtront comme par exemple les onglets suivants:



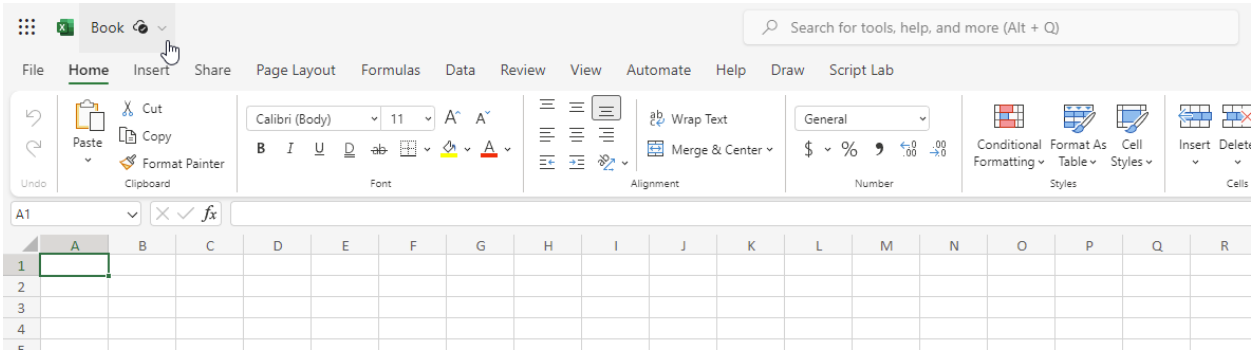
8 Enregistrement de Scripts (ie macros)

Il n'existe pas à ce jour et à notre connaissance de convertisseur de VBA en Office Script!!!

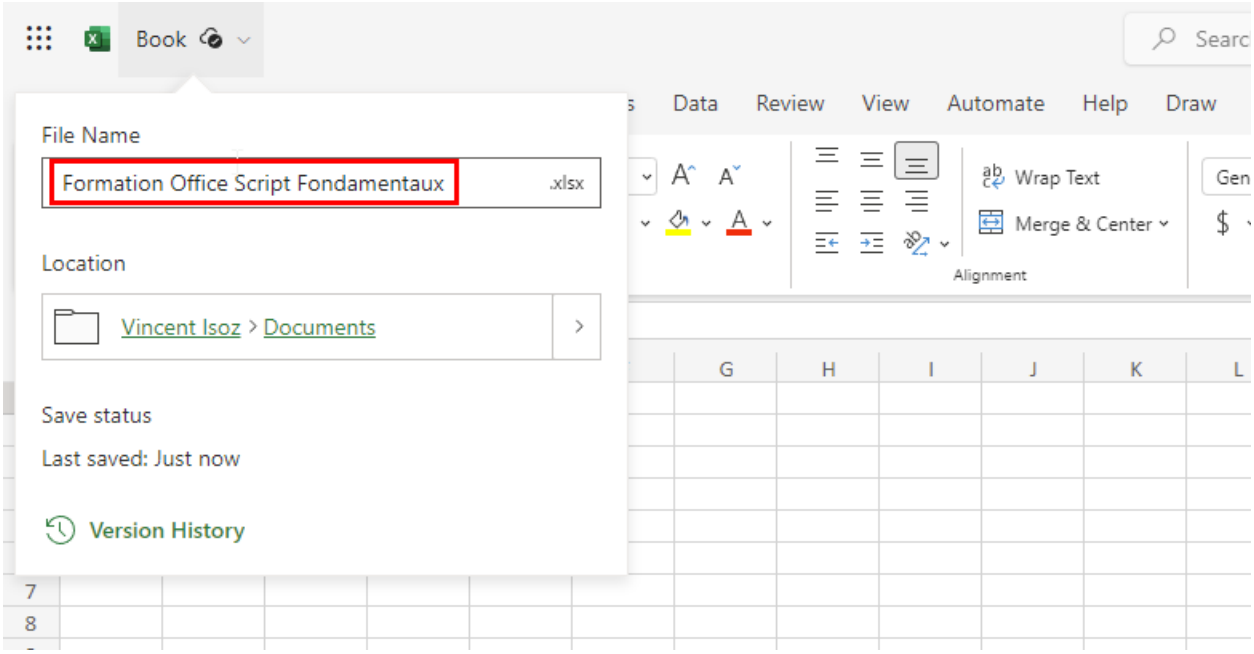
Pour découvrir l'enregistrement de scripts dans Microsoft Excel, ouvrez la version Online d'Excel et ensuite cliquez sur *Blank workbook*:



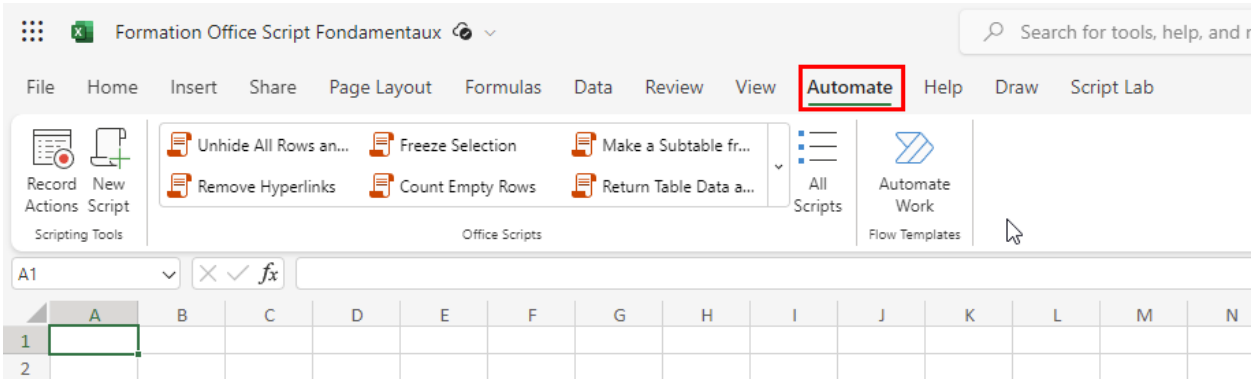
Nous avons alors:



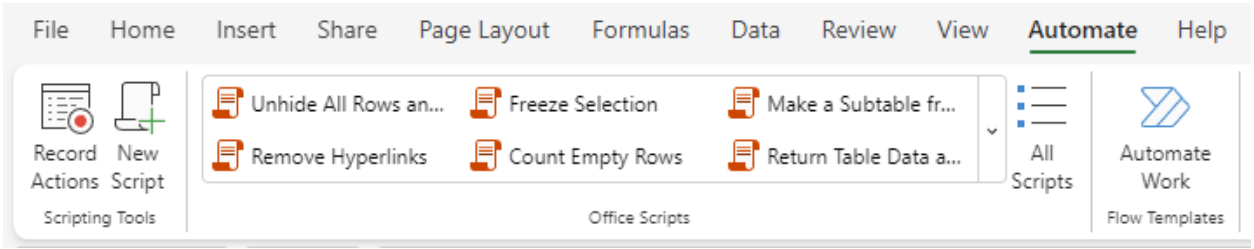
Enregistrez le fichier par exemple avec le nom *Formation Office Script Fondamentaux*:



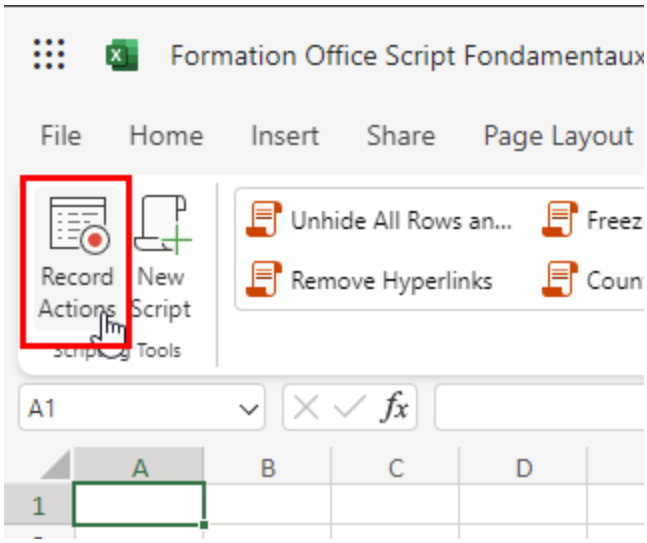
Cliquez ensuite sur l'onglet **Automate**:



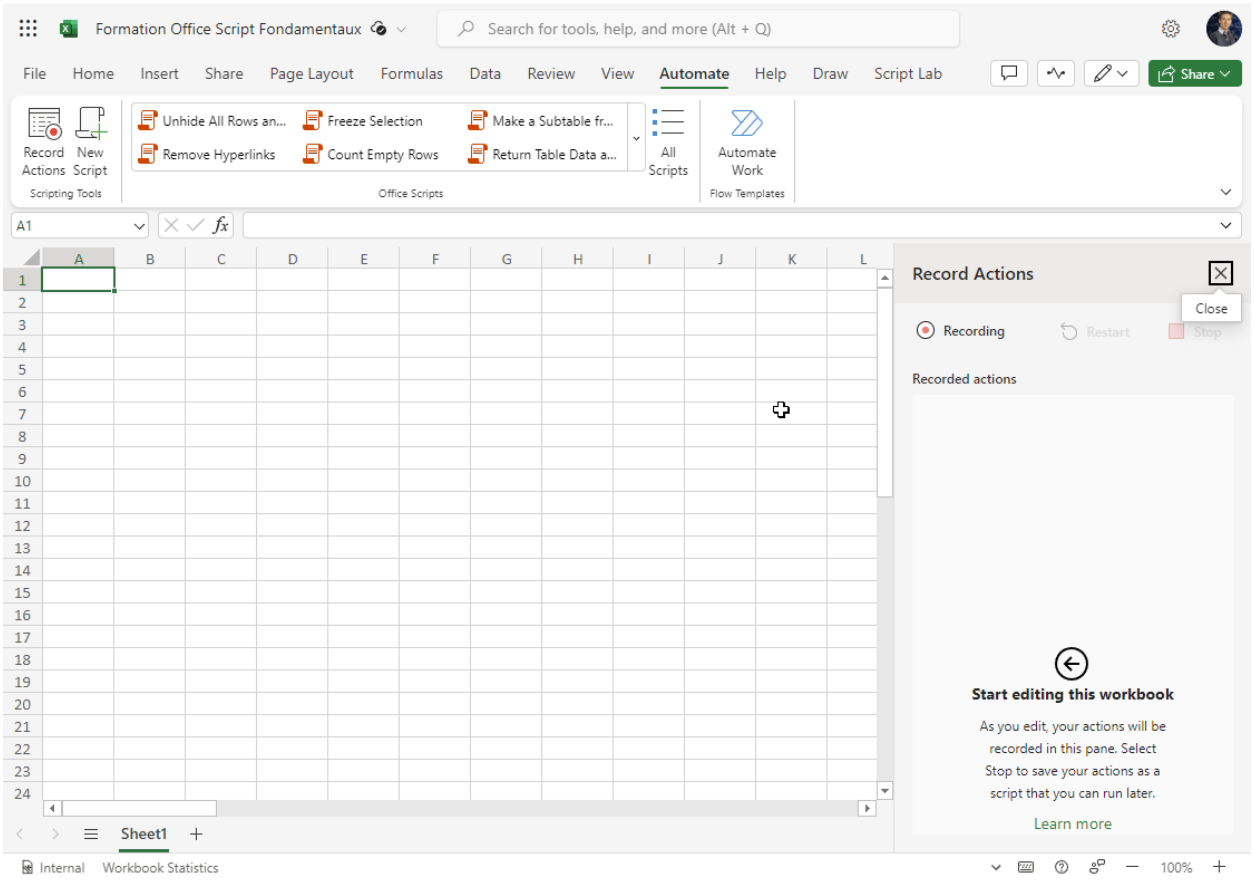
Apparaît alors à l'écran:



Cliquez sur **Record Actions**:

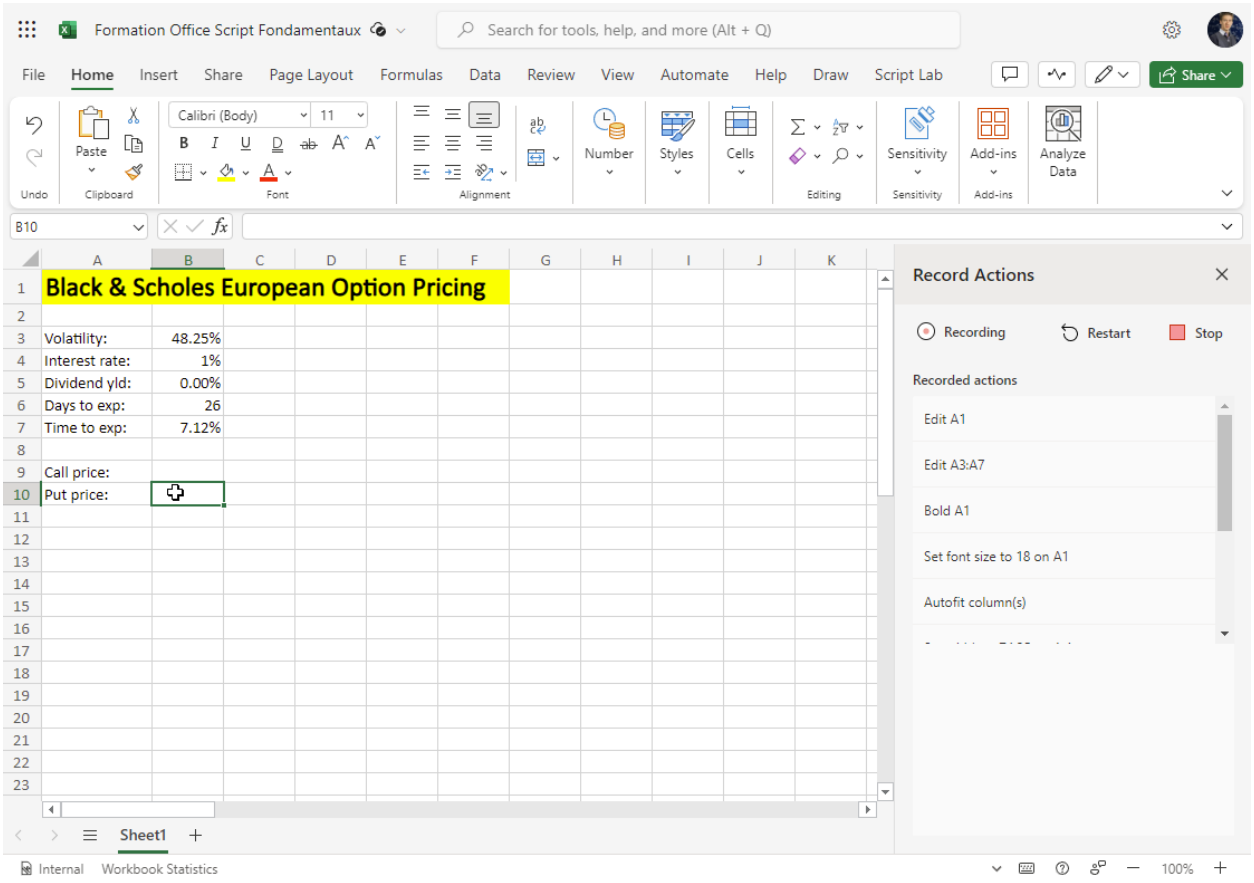


Un panneau s'ouvre à droite pour indiquer que l'enregistrement est en cours:

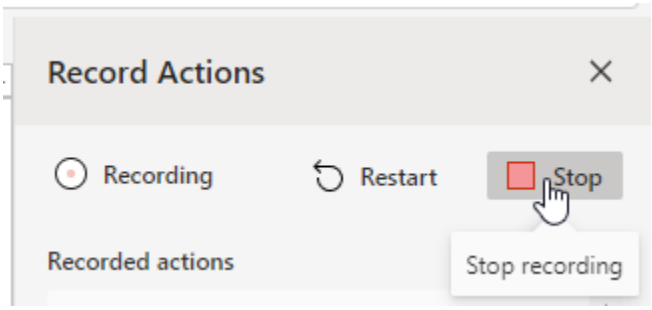


8.1 Première macro de type formatage et saisie seuls (sans bugs/sans pièges)

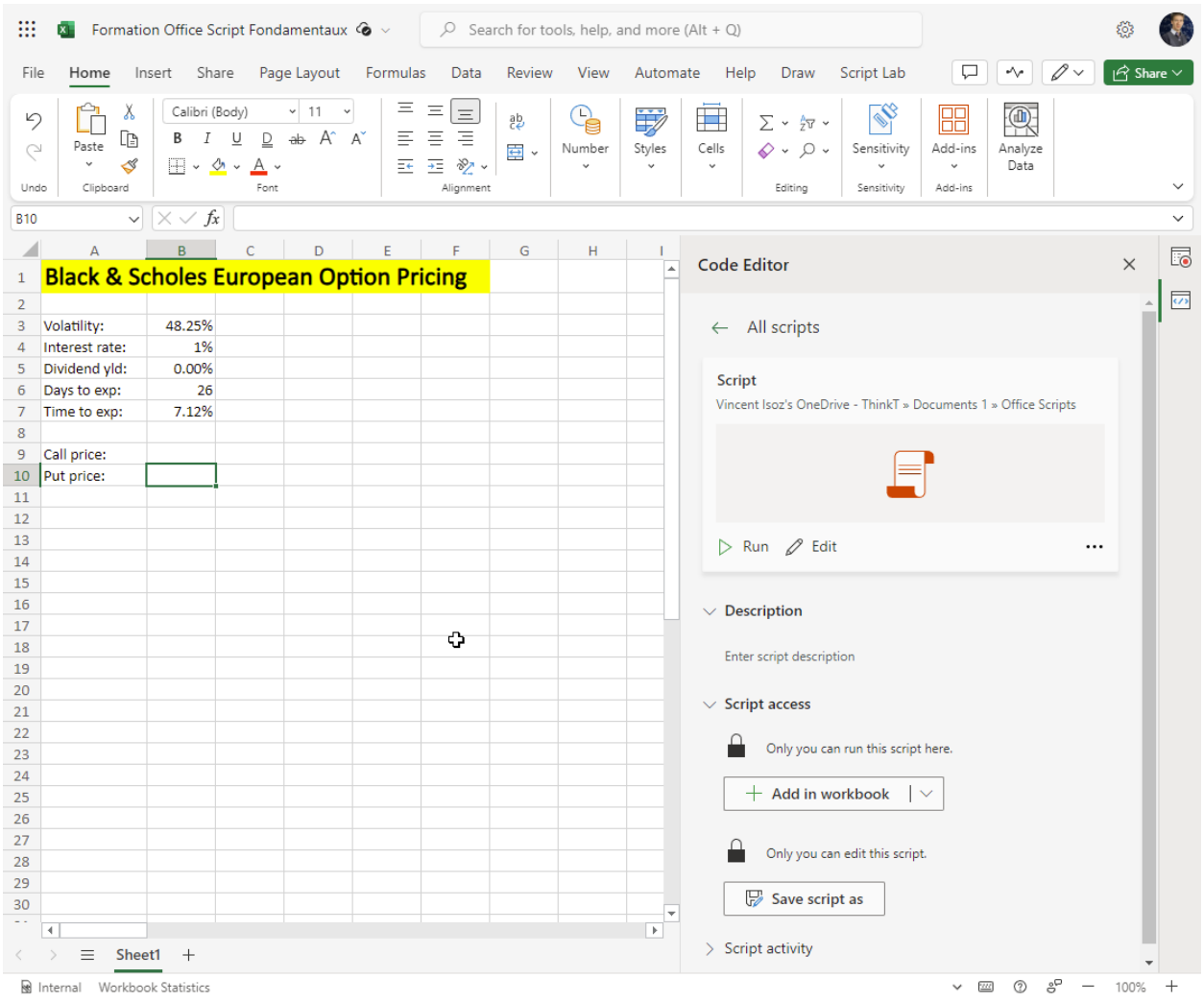
Créez pendant l'enregistrement la petite mise en page avec les données suivantes:



Une fois terminé, cliquez sur **Stop**:



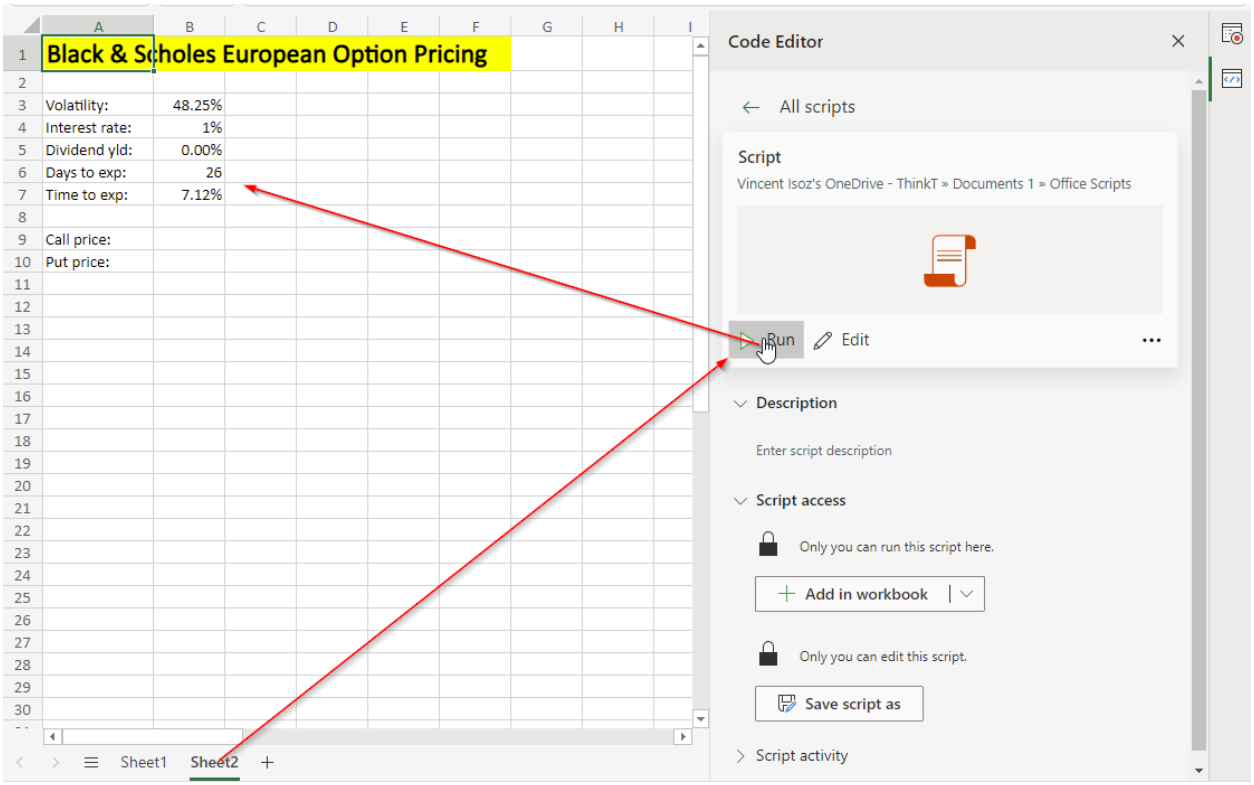
Nous avons alors:



Avec le VBA, dans le cas par défaut, les macros étaient enregistrées dans le classeur où elles étaient créés. Avec Office Script, la macro est disponible, visible et exécutable dans tous vos classeurs Microsoft Excel (cela devient alors très vite un chaos généralisé…).

8.1.1 Exécuter le script

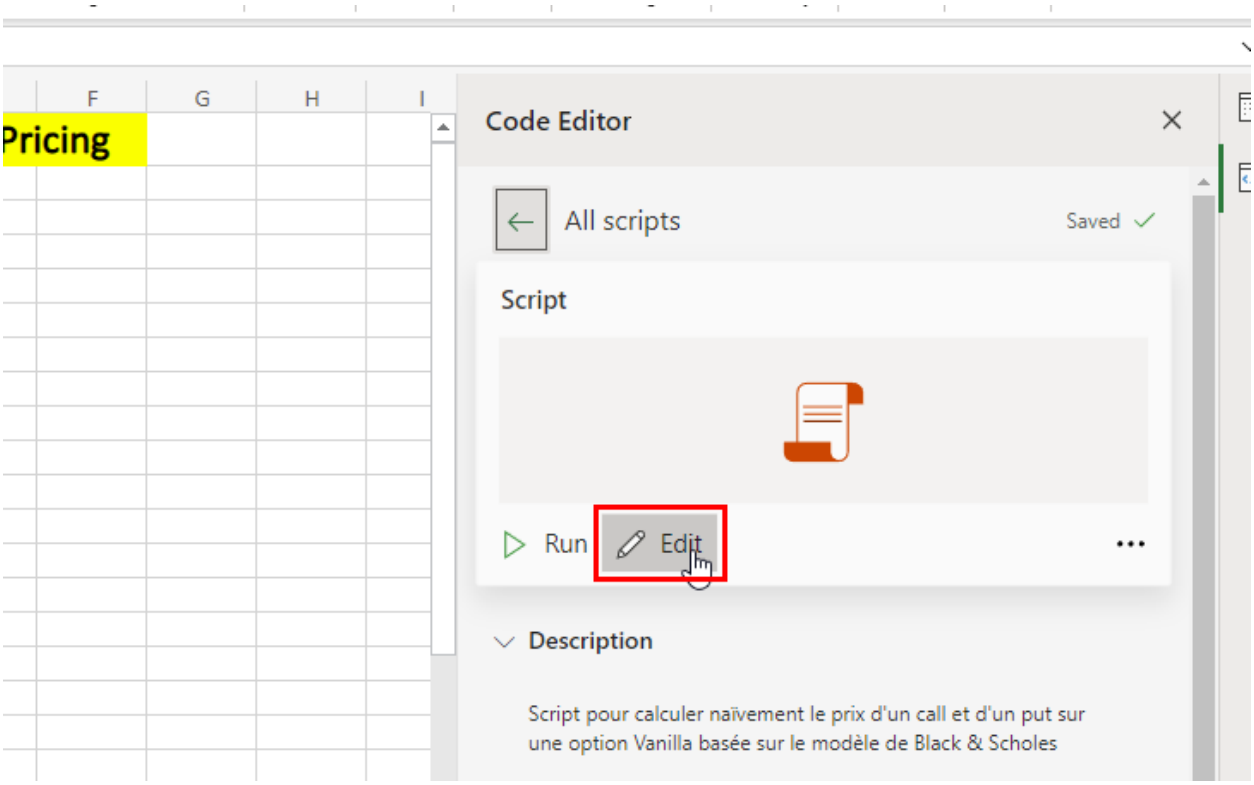
Pour exécuter le script juste après avoir enregistré une macro office script, vous cliquez sur **Run** typiquement dans une nouvelle feuille:



8.1.2 Éditer le script

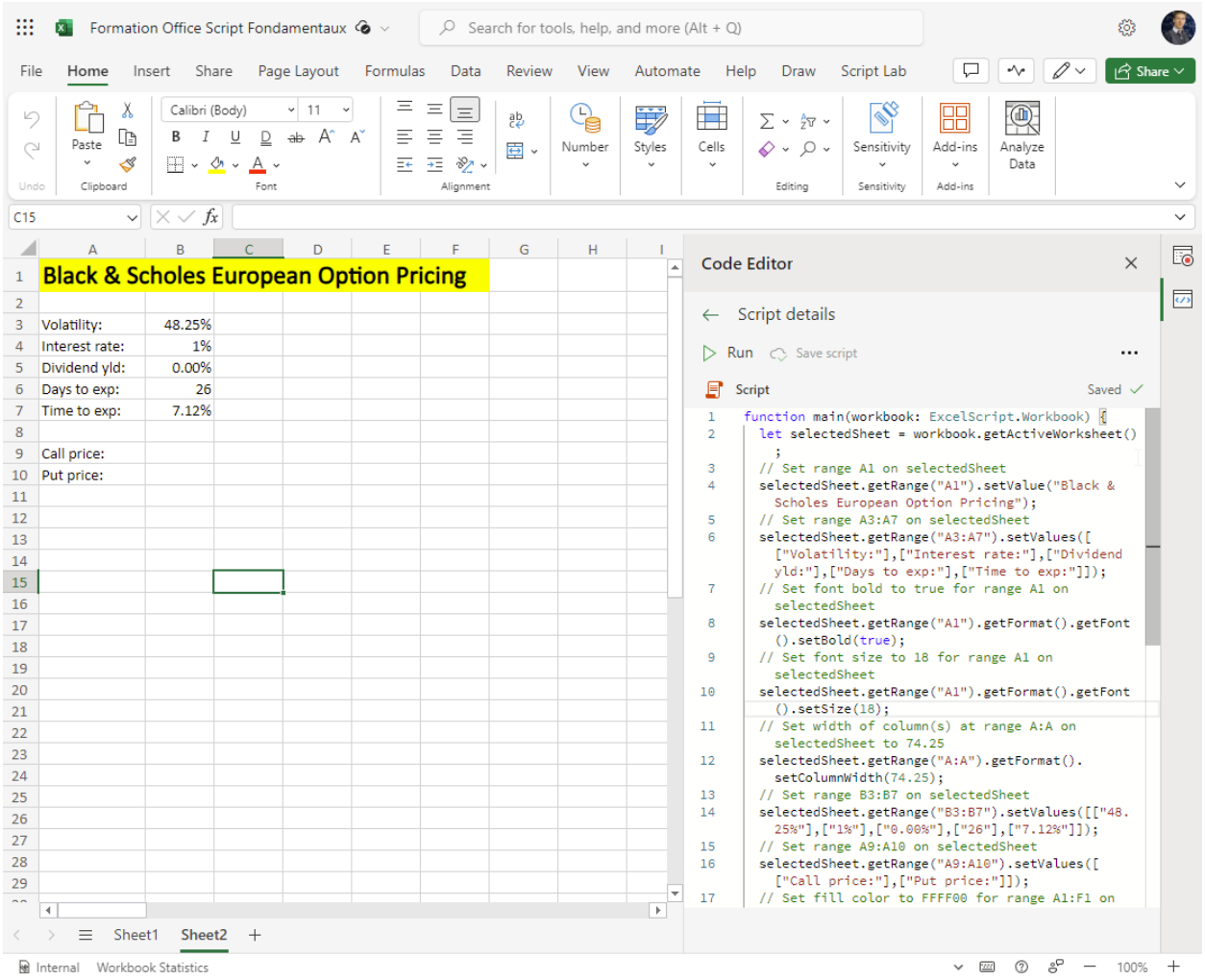
Public Internal

Pour éditer le script de la macro que nous venons d'enregistrer, nous cliquons sur le bouton **Edit**:



Nous verrons alors à l'écran:

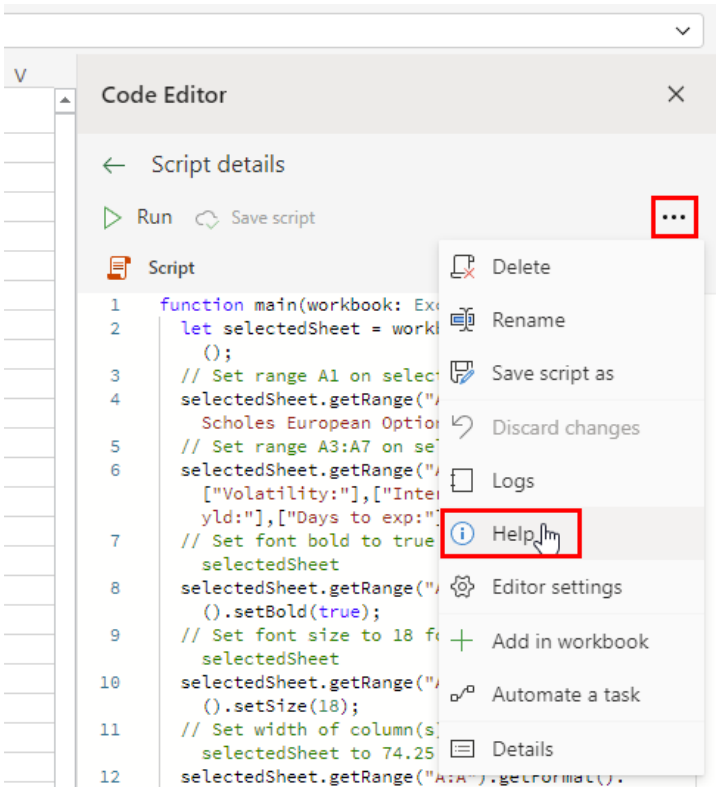
Public Internal



Il est facile de faire des changements sur le script sur la partie droite si on a des notions de l'anglais.

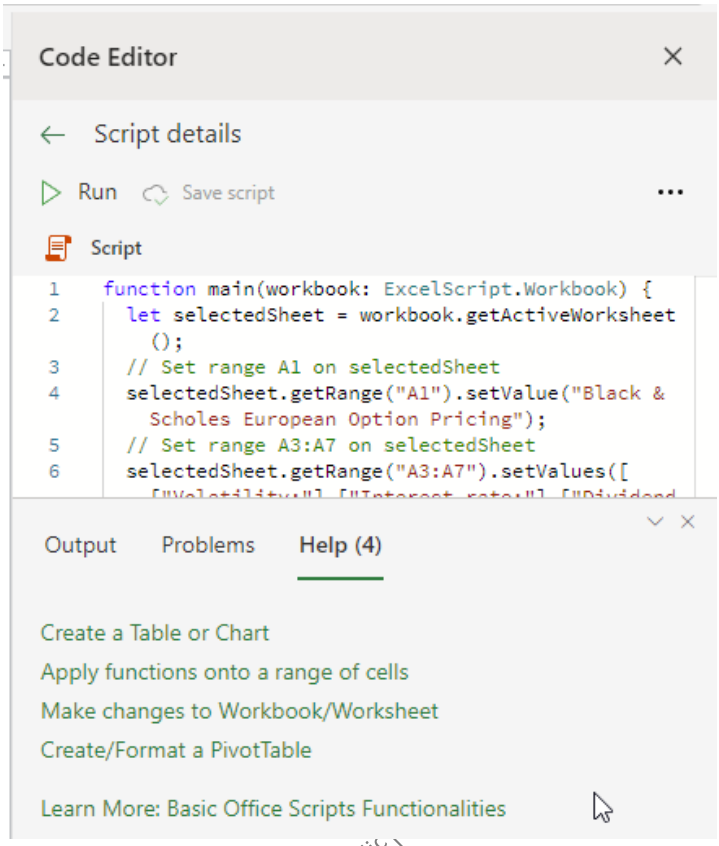
8.1.2.1 Aide

Si nous allons dans le menu de script et cliquons sur **Help**:



Nous avons alors:

Public
Internal

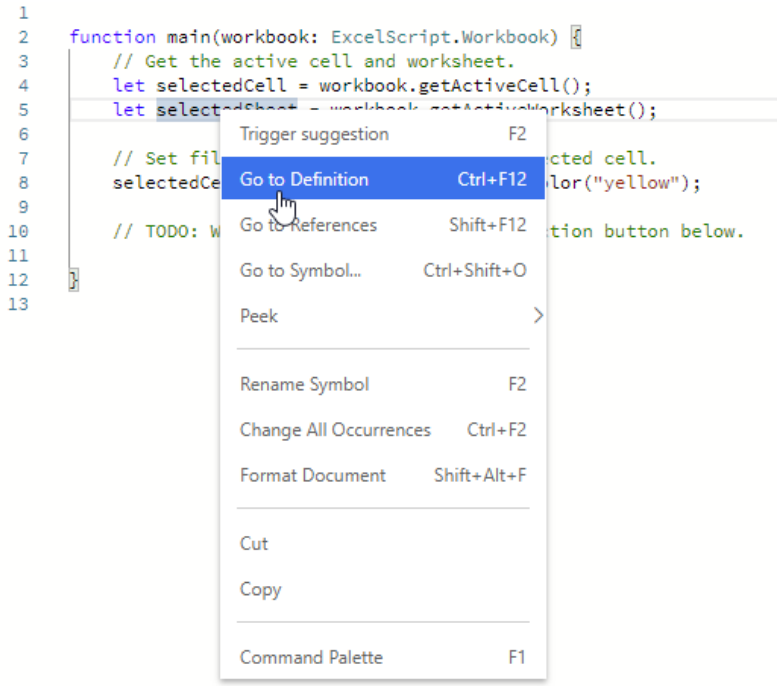


Public
External

Malheureusement c'est donc relativement inutile… Le moins pire est d'aller sur:

https://learn.microsoft.com/en-us/javascript/api/office-scripts/overview?view=office-scripts

Nous avons aussi au clic droit quelques options:

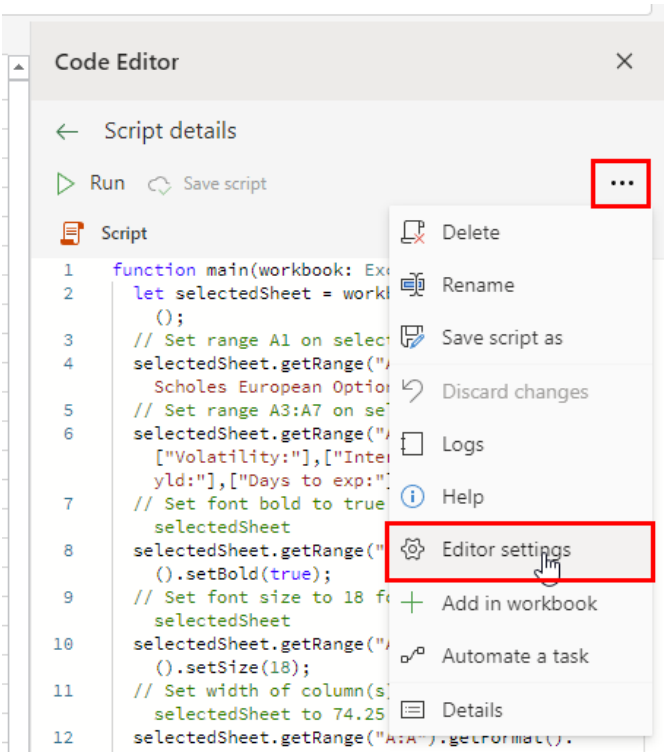


Nous avons quelques options qu'il suffit de parcourir pour comprendre. Certaines sont relativement inutiles à ce jour.

Public
Internal

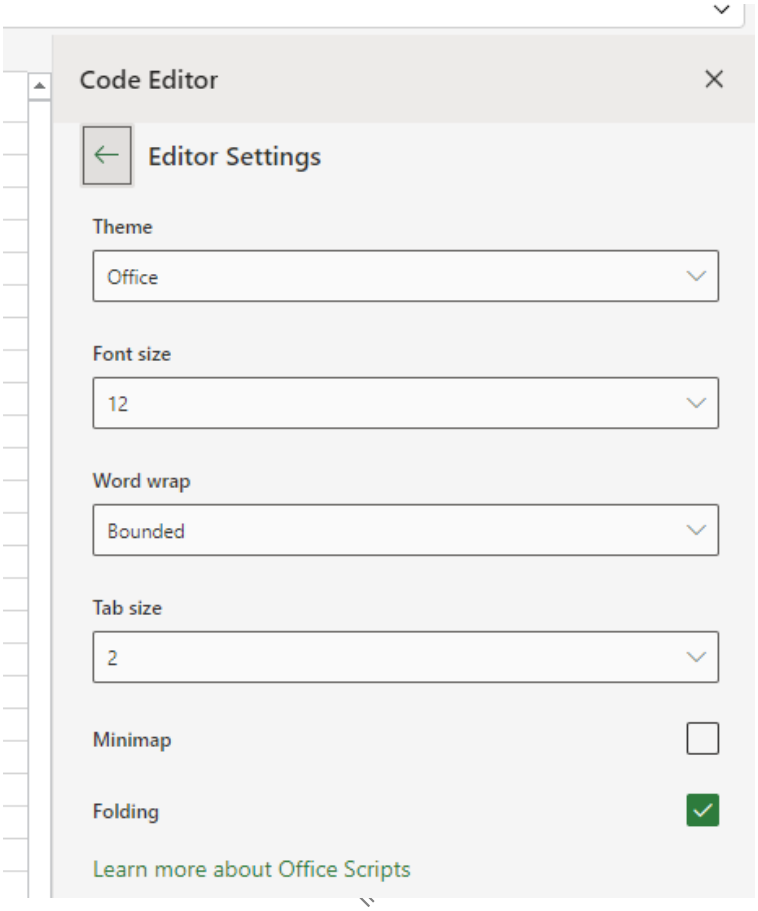
8.1.2.2 Configurer l'éditeur

Il existe certaines options de configuration de l'éditeur (bien que nous soyons très très loin des possibilités de l'éditeur Visual Basic):

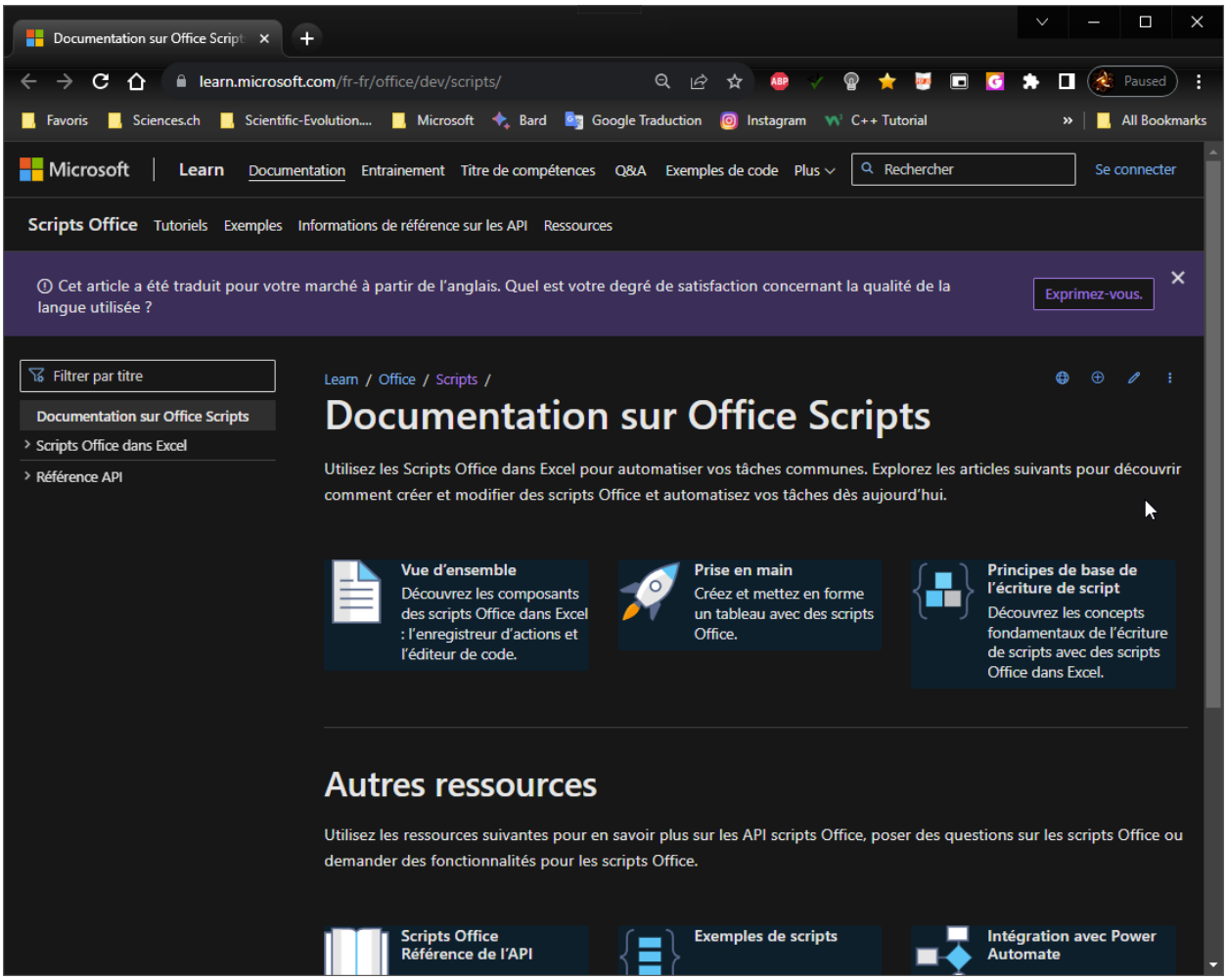


Nous avons alors:

Public
Internal



Si nous cliquons sur l'option *Learn more about Office Scripts*, nous avons:

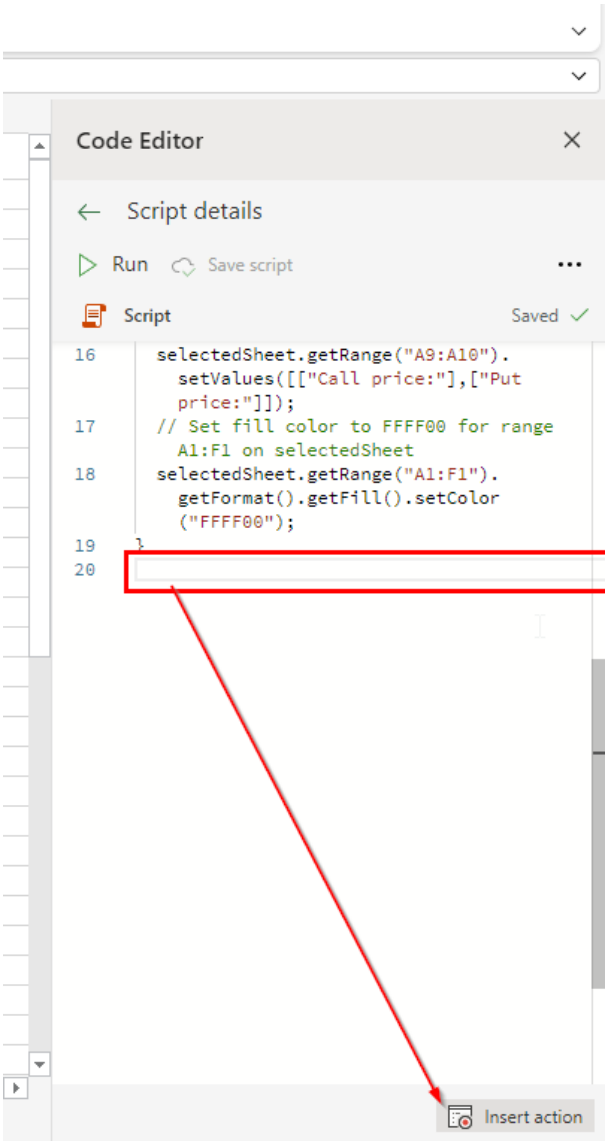


Donc bof rien de spécial…

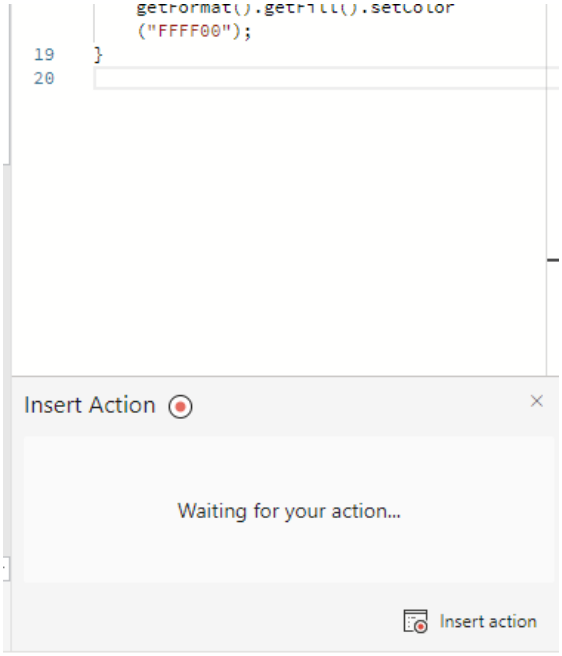
8.1.2.3 Ajouter une action enregistrée

Pouvons nous rajouter une action enregistrée a un script existant? Nous savons que cela n'est pas possible avec les macros VBA! Mais nous nous allons voir que c'est possible avec les Office Script!

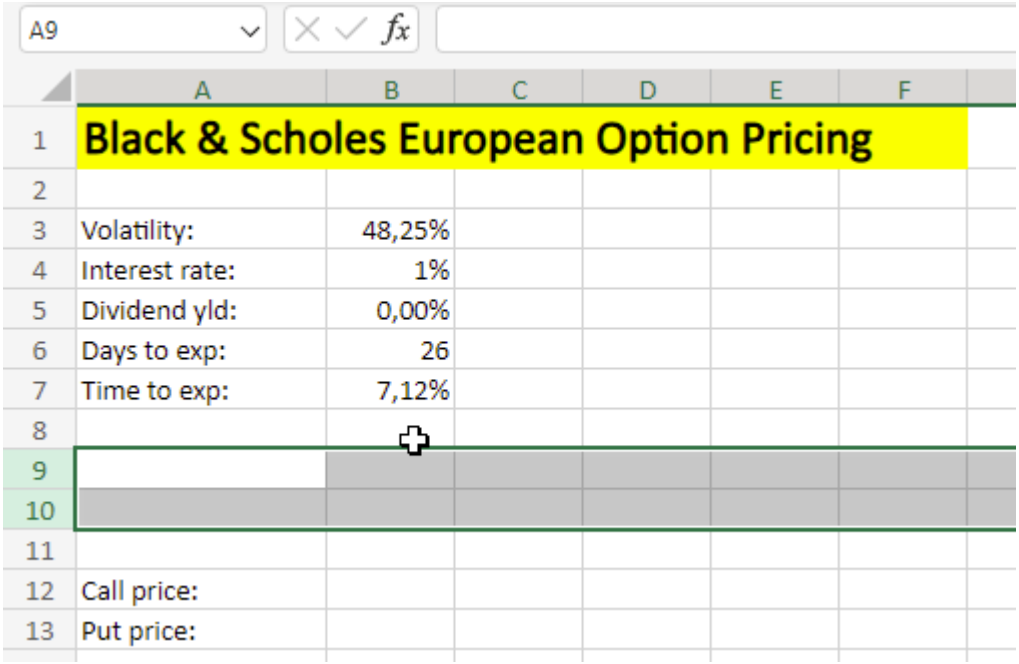
Après avoir cliqué a l'extérieur des accolades du code existant (voir capture ci-dessous), nous voyons un bouton **Insert action**:



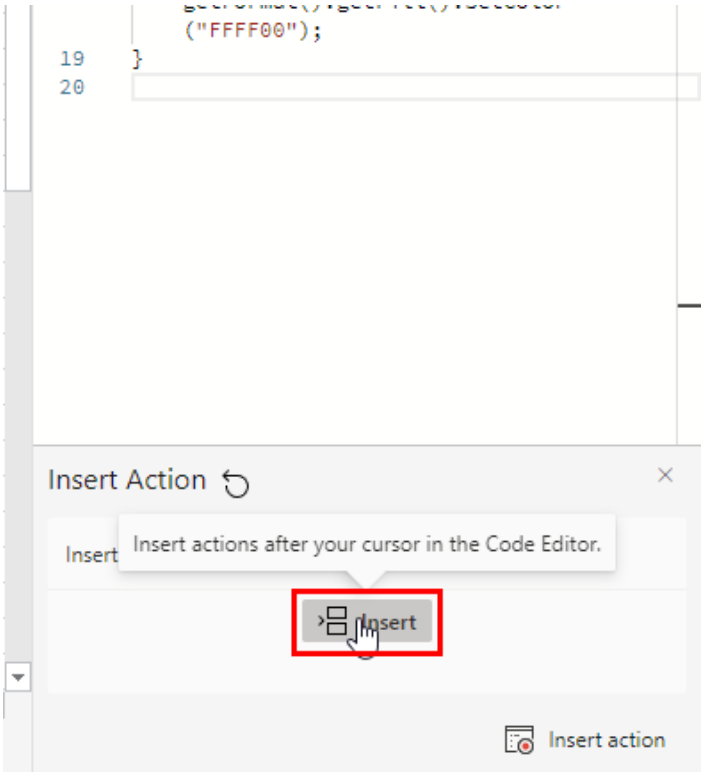
Une fois cliqué, nous avons:



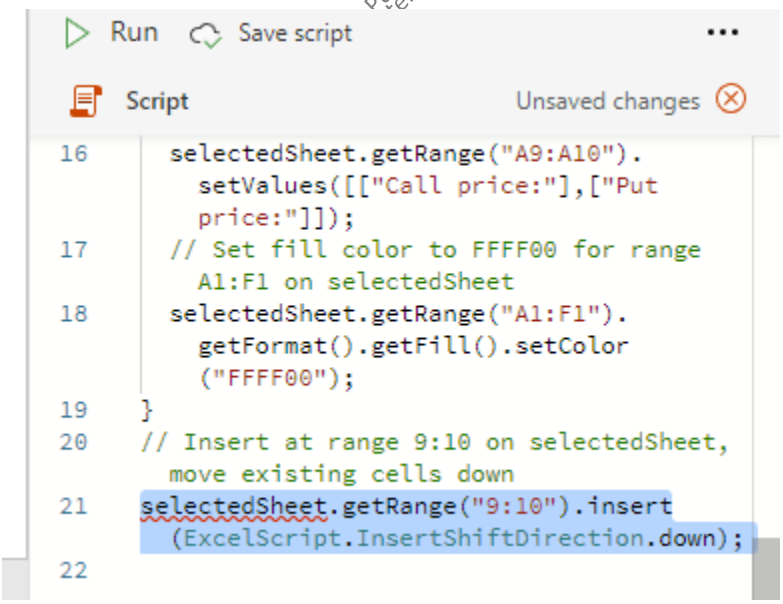
Rajoutons par exemple deux lignes à notre mise en page:



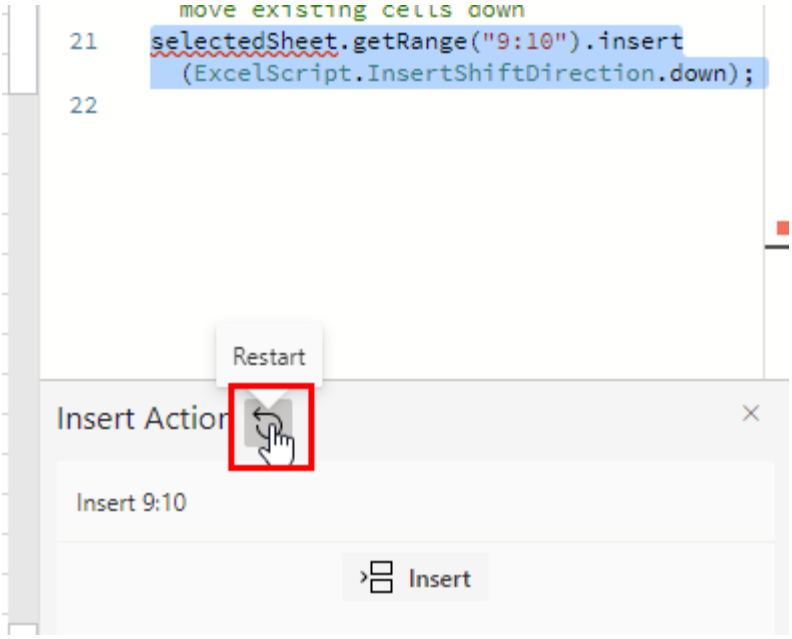
Nous voyons alors un bouton **Insert** apparaître



Si nous cliquons dessus, nous avons alors:



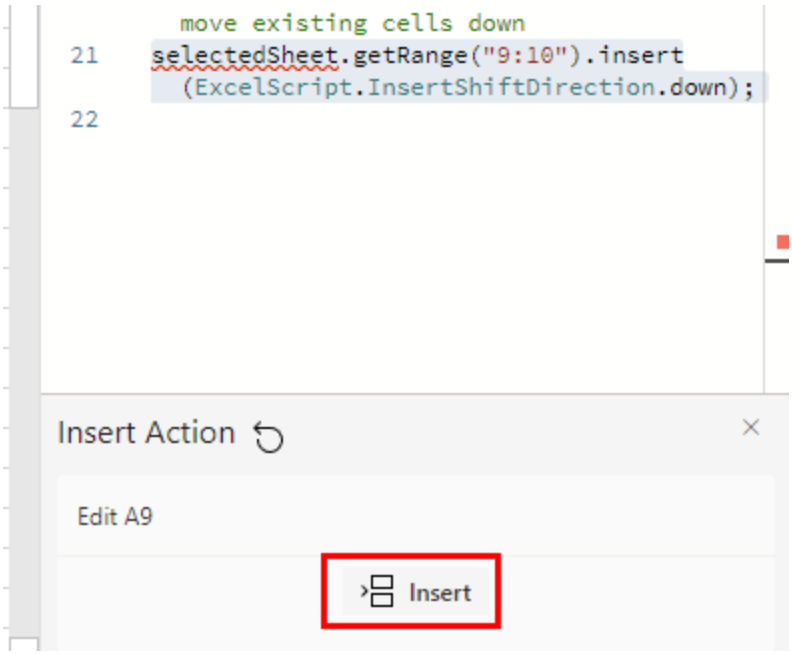
Ensuite comme on ne peut pas ajouter plusieurs actions d'un coup on clique sur **Restart**:



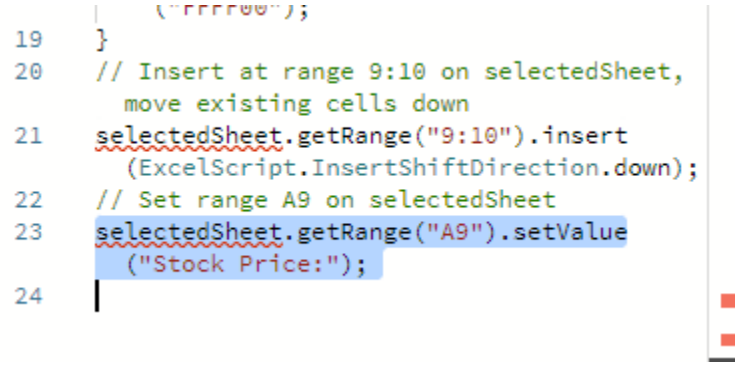
Nous écrivons *Stock Price:* et *Strike Price*: comme visible ci-dessous:

7	Time to exp:	7,12%	
8			
9	Stock Price:		
10	Strike Price:		
11			
12	Call price:		
13	Put price:		

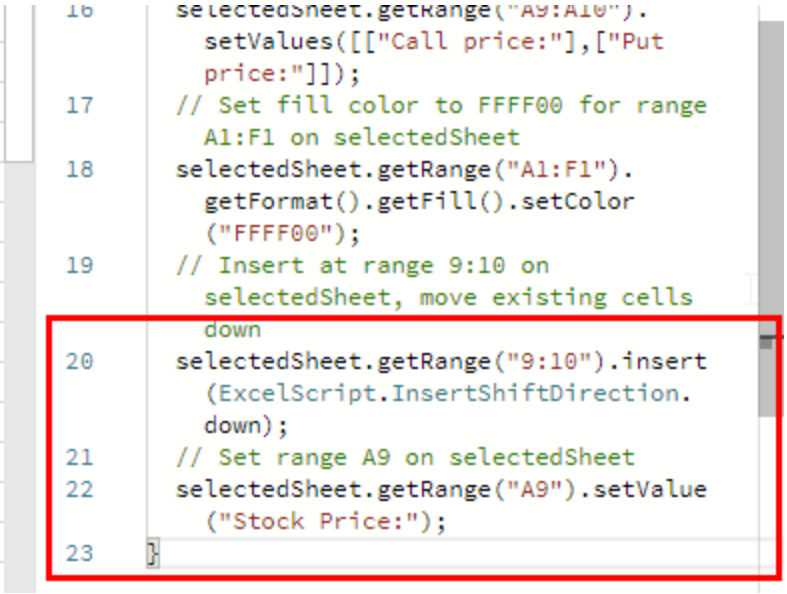
Et à nouveau sur **Insert**:



Ce qui donne:



Le code est rouge parce qu'il n'est pas dans la fonction *main*, nous faisons alors un couper/coller pour le placer au bon endroit:



```
16    selectedSheet.getRange("A9:A10").
        setValues([["Call price:"],["Put
        price:"]]);
17    // Set fill color to FFFF00 for range
        A1:F1 on selectedSheet
18    selectedSheet.getRange("A1:F1").
        getFormat().getFill().setColor
        ("FFFF00");
19    // Insert at range 9:10 on
        selectedSheet, move existing cells
        down
20    selectedSheet.getRange("9:10").insert
        (ExcelScript.InsertShiftDirection.
        down);
21    // Set range A9 on selectedSheet
22    selectedSheet.getRange("A9").setValue
        ("Stock Price:");
23  }
```

8.1.3 Aide

Il existe plusieurs aides pour apprendre l'office script mais nous sommes très loin des fonctionnalités de l'éditeur VBA malheureusement

8.1.3.1 Aide en ligne de Microsoft

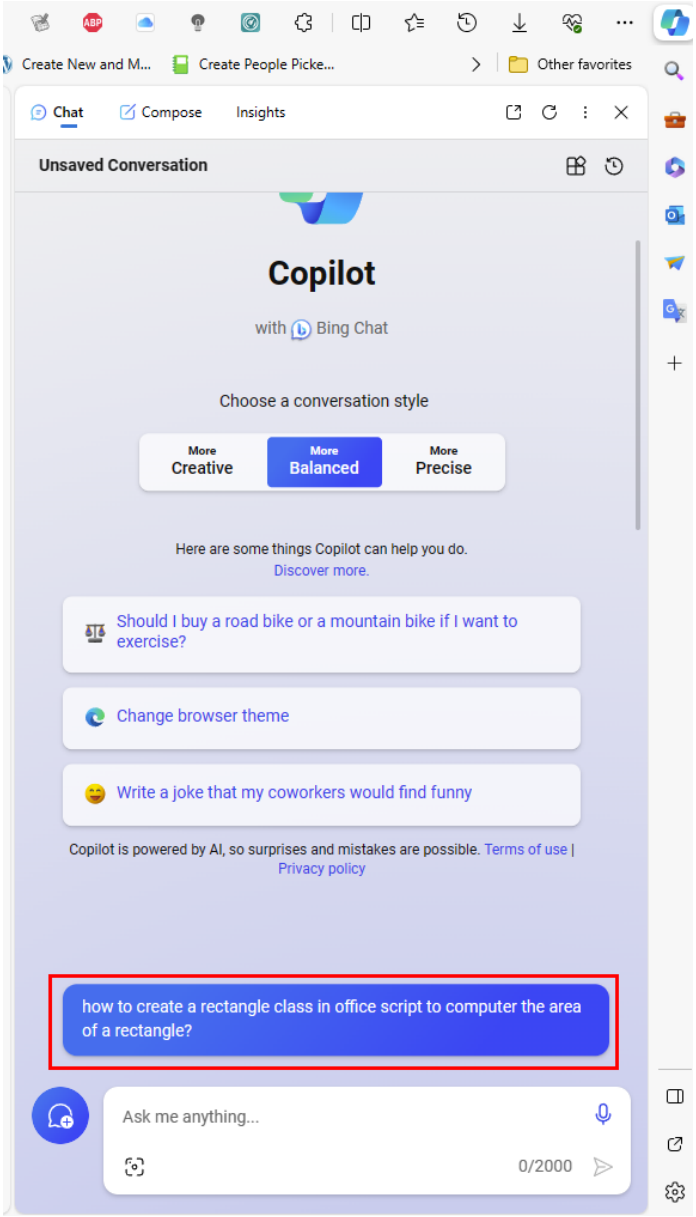
Public Internal

D'abord, outre les liens déjà donnés dans la section Médiagraphie, rappelons la documentation d'environ 1'100 pages A4 de documentation disponible ici:

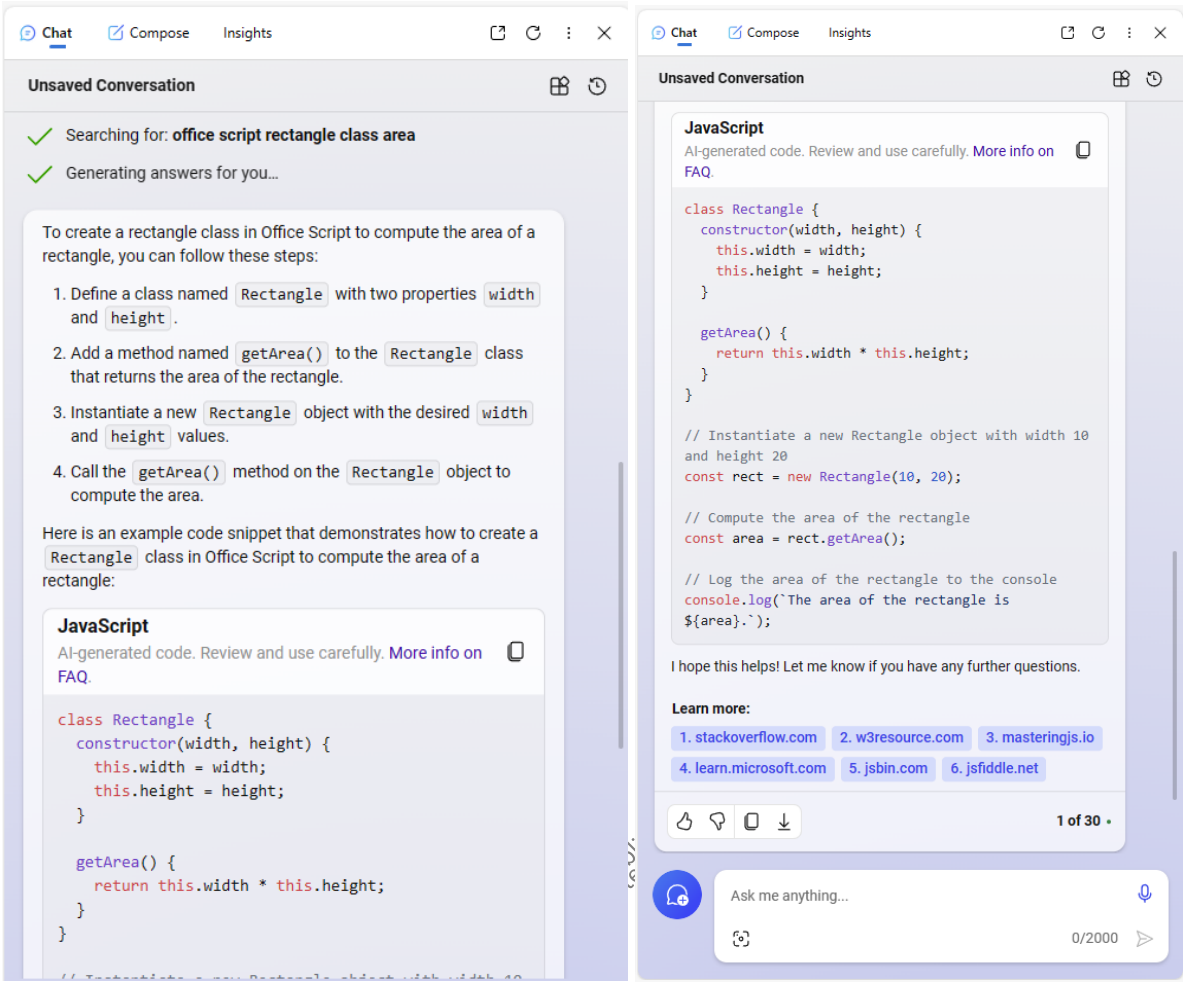
https://learn.microsoft.com/en-us/javascript/api/office-scripts/excelscript?view=office-scripts

8.1.3.2 Utilisation de ChatGPT / Bard

Évidemment depuis 2022 les LLM changent totalement la donne. Voyons un exemple avec Copilot:



La réponse pour ce cas particulier est:



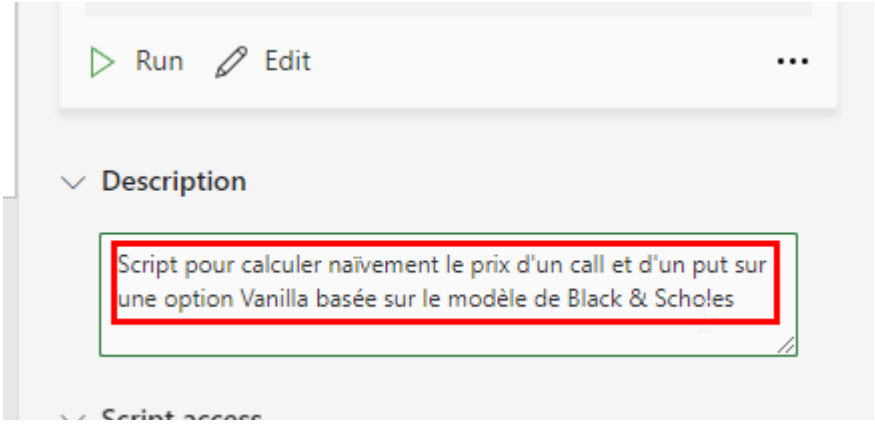
Ce qui est pas mal du tout puisque pleinement fonctionnel!

8.1.4 Options d'affichage

Nous avons déjà traité le sujet à la page 36.

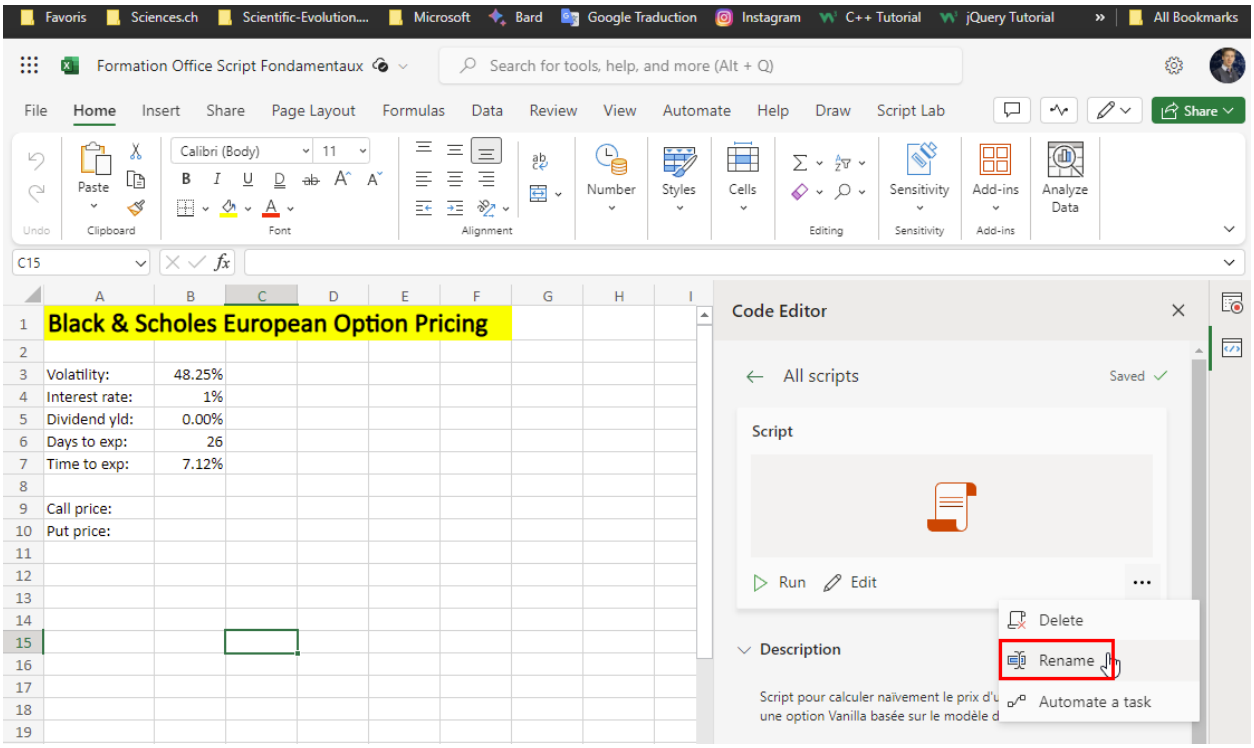
8.1.5 Décrire le script

Dans le panneau de Script il y a un champ **Description** qui permet de faire ce que son nom décrit…:

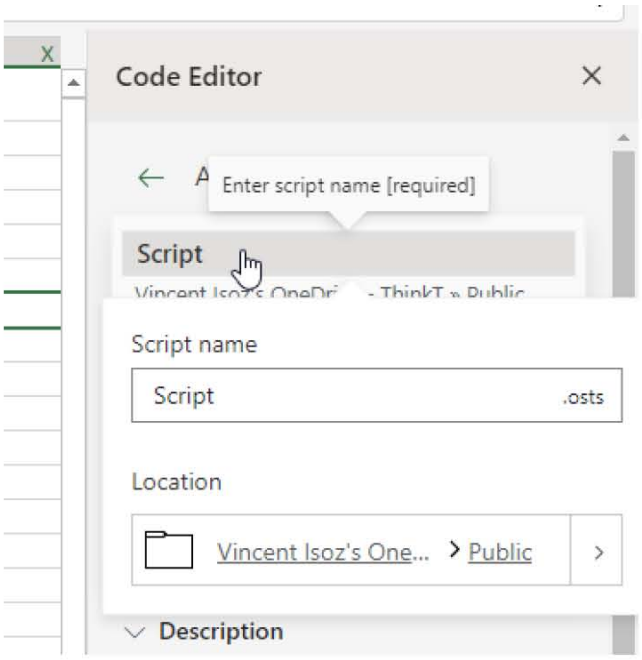


8.1.6 Renommer le script

Pour renommer le script c'est très simple. Il suffit d'aller dans le menu des options … :



On peut aussi renommer en cliquant sur le nom du script:



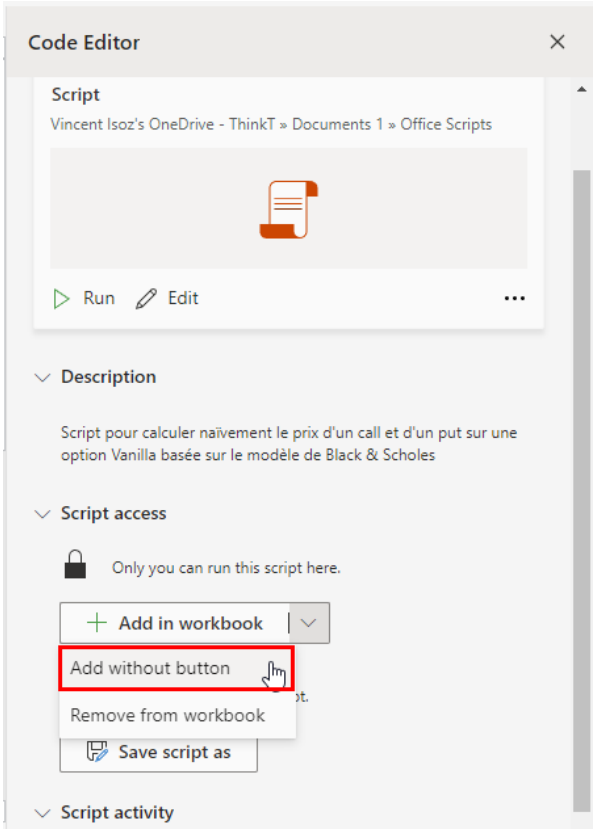
Public
Internal

8.1.7 Rendre la macro accessible

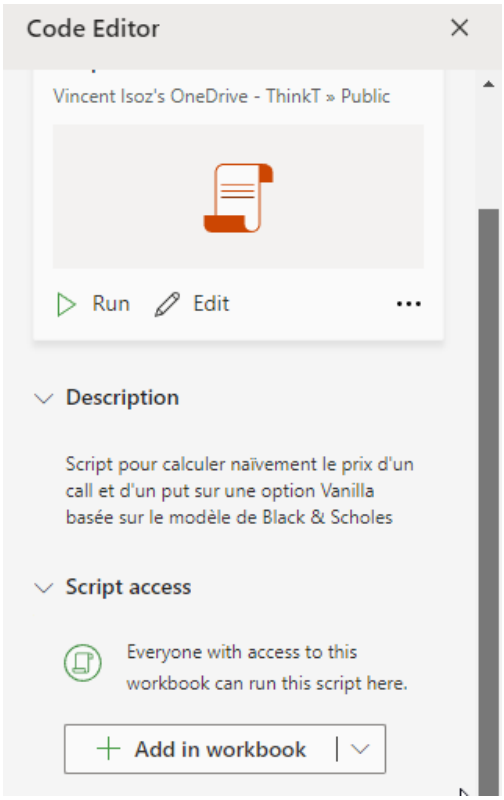
Nous avons déjà mentionné les nombreuses limites actuelles d'interfaçage des macros Office Script. Voyons donc le peu d'options disponibles…

8.1.7.1 Sans bouton

La première option consiste à cliquer sur l'option **Script access** et de cliquer sur *Add without button*:

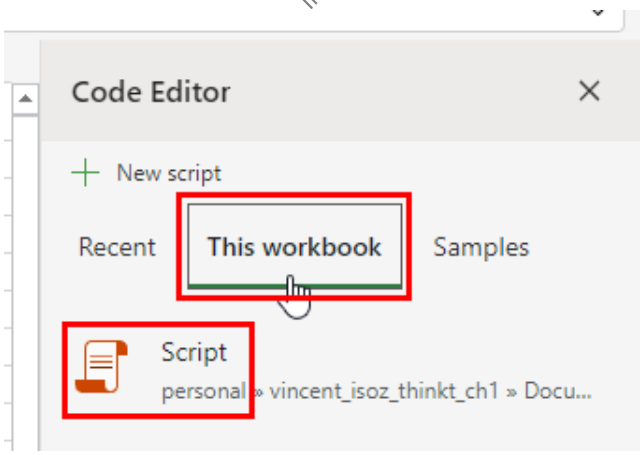


Si nous cliquons sur cette option, nous avons:



Nous avons alors dans l'onglet **This workbook**:

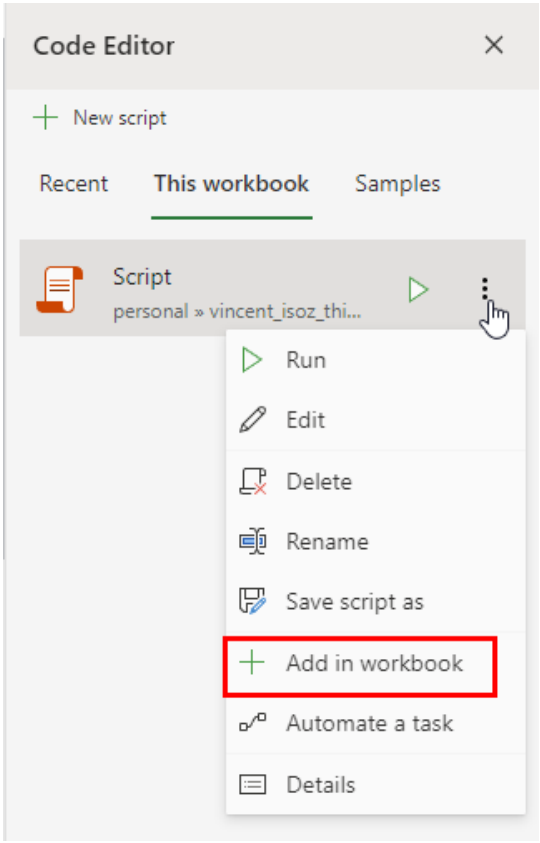
Public
Internal



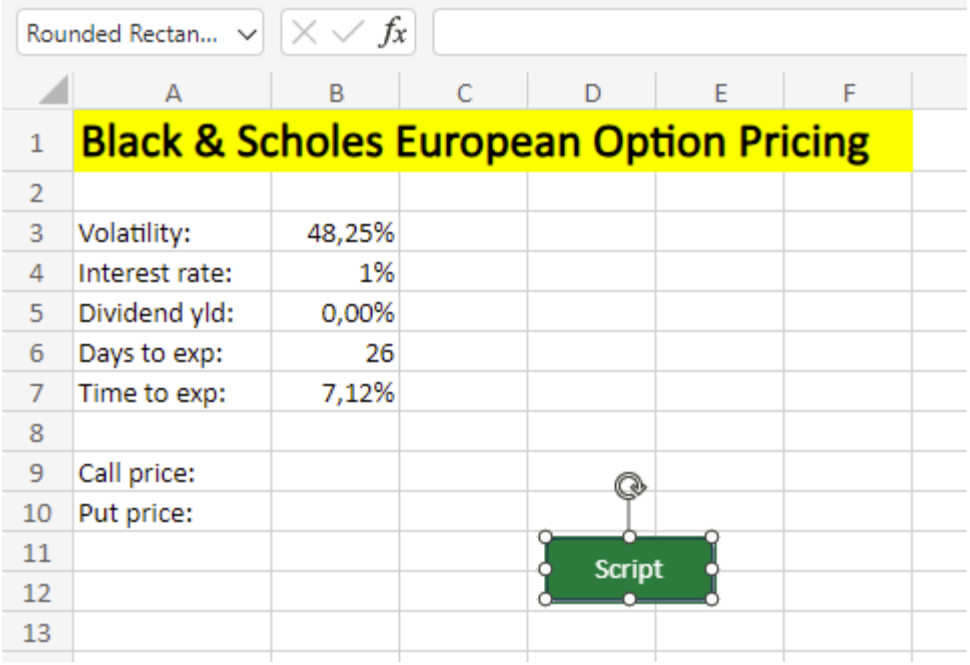
Mais il sera aussi disponible dans les autres classeur via l'onglet **Automate**.

8.1.7.2 Avec bouton

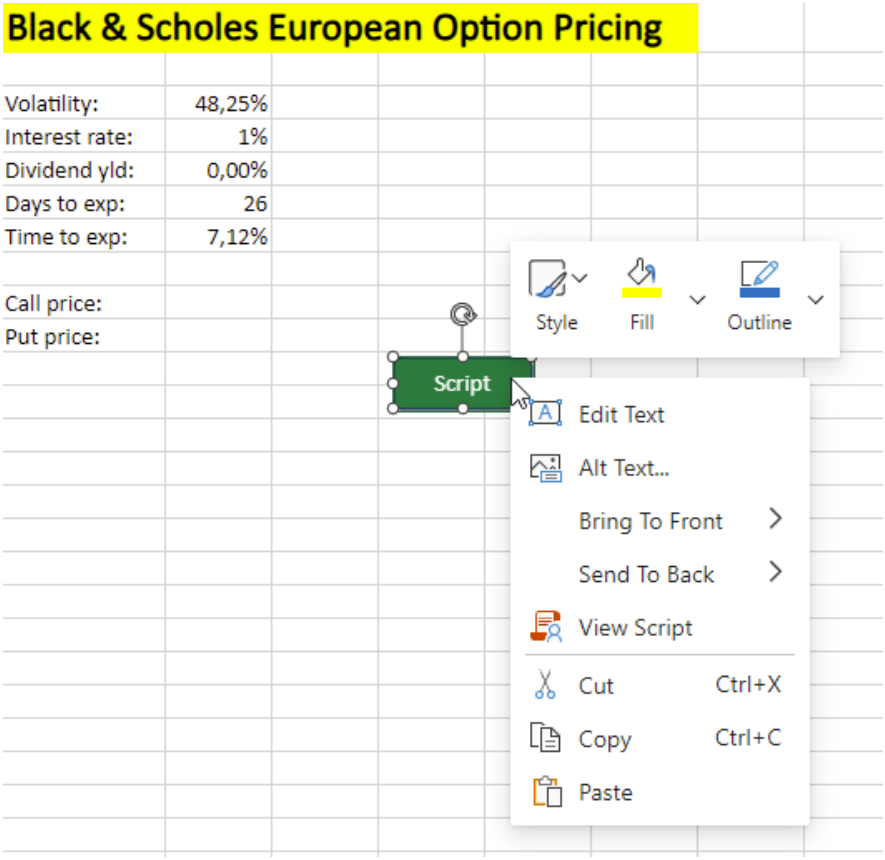
Si nous allons maintenant par exemple dans le menu du script et nous cliquons sur **Add in workbook**:



Cela donne alors:



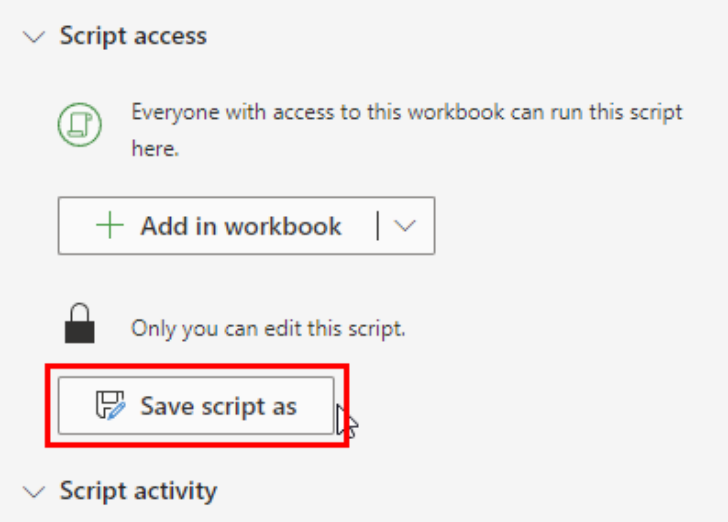
Le bouton est renommable mais pas contre pas supprimable avec la touche DEL du clavier (il faut couper le bouton pour l'enlever):



Avec les outils de dessins habituels de Microsoft Excel, nous pouvons changer la forme et les effets du bouton selon les goûts…

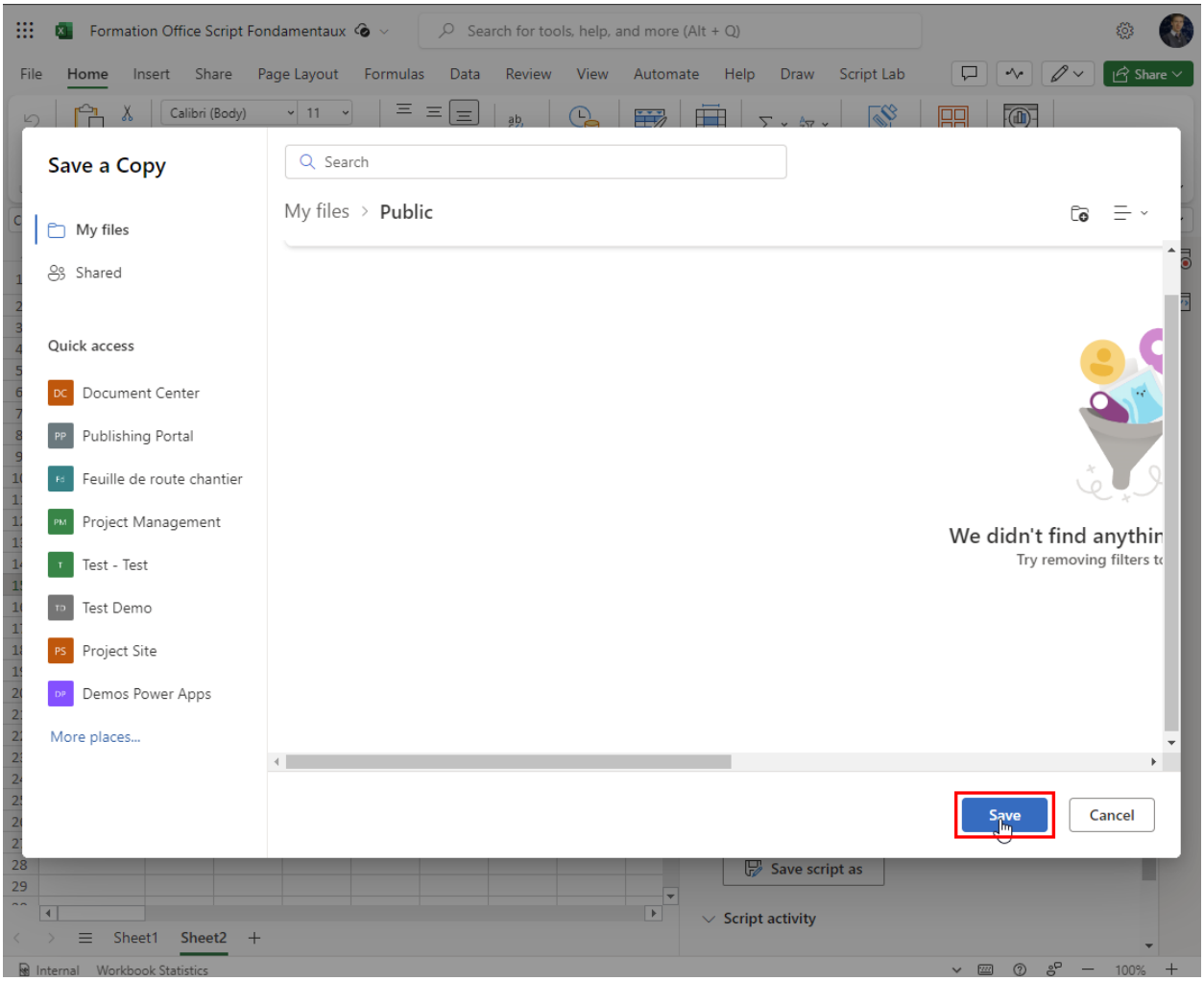
8.1.8 Sauvegarder le script (et le rendre éditable par d'autres)

Pour cela, nous cliquons sur **Save script as**:

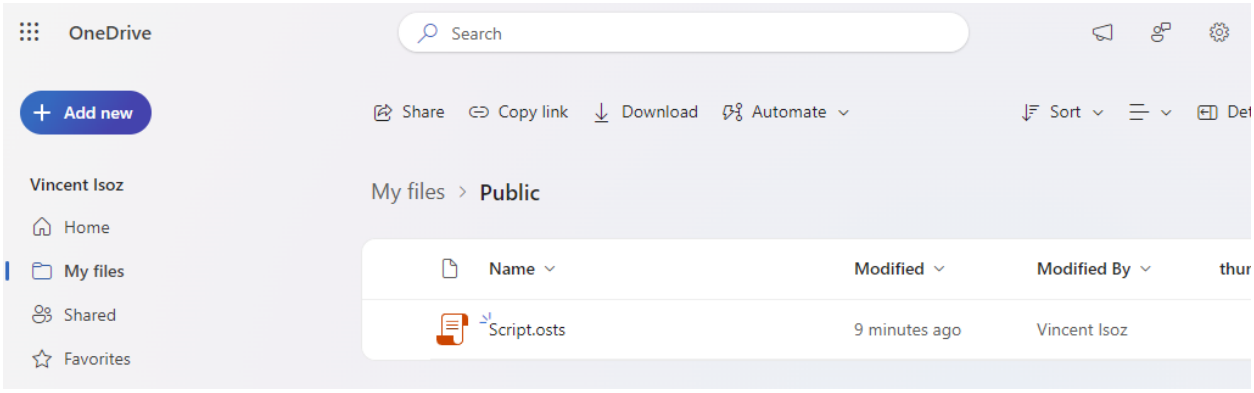


Nous avons alors:

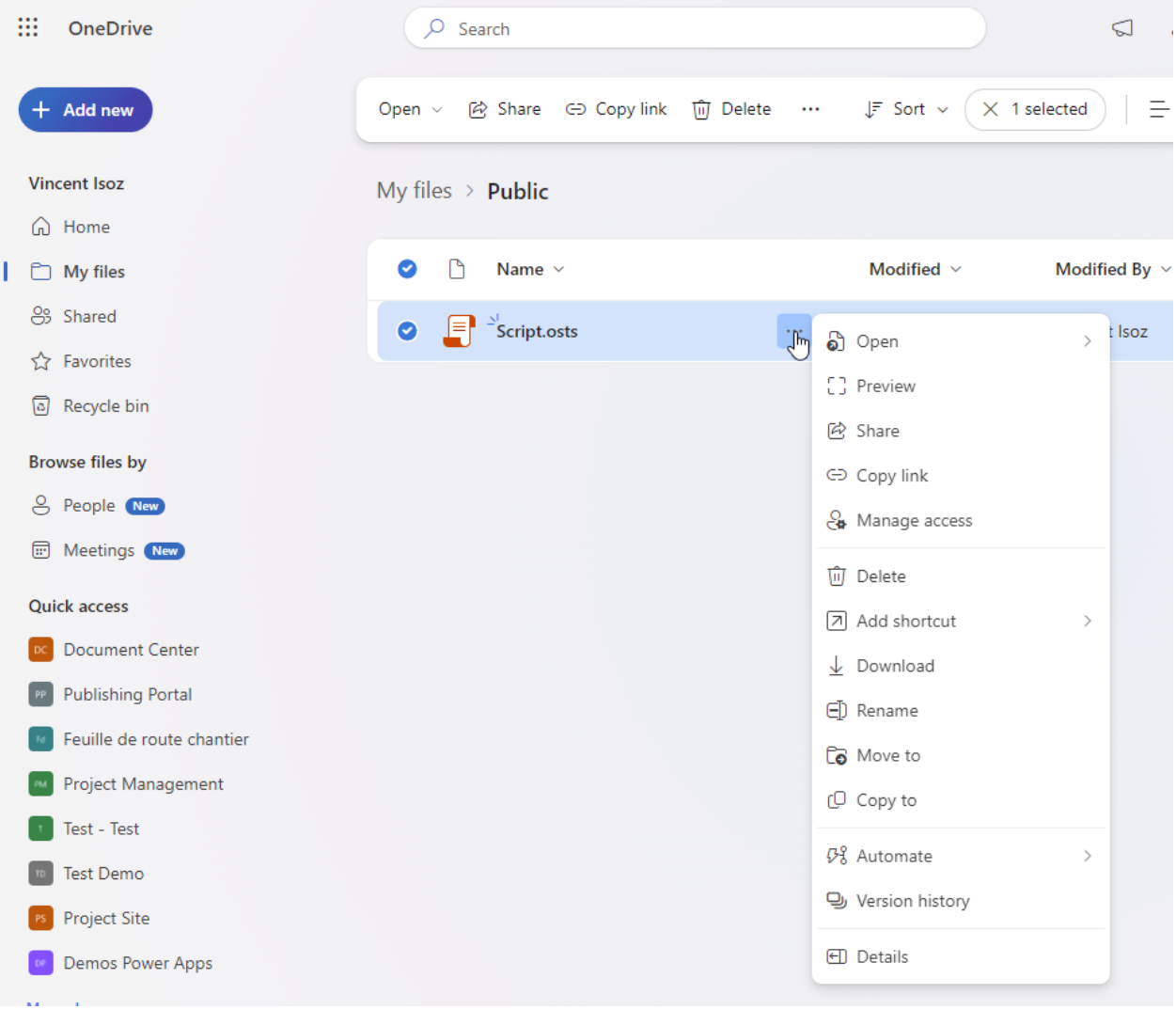
Public
Internal



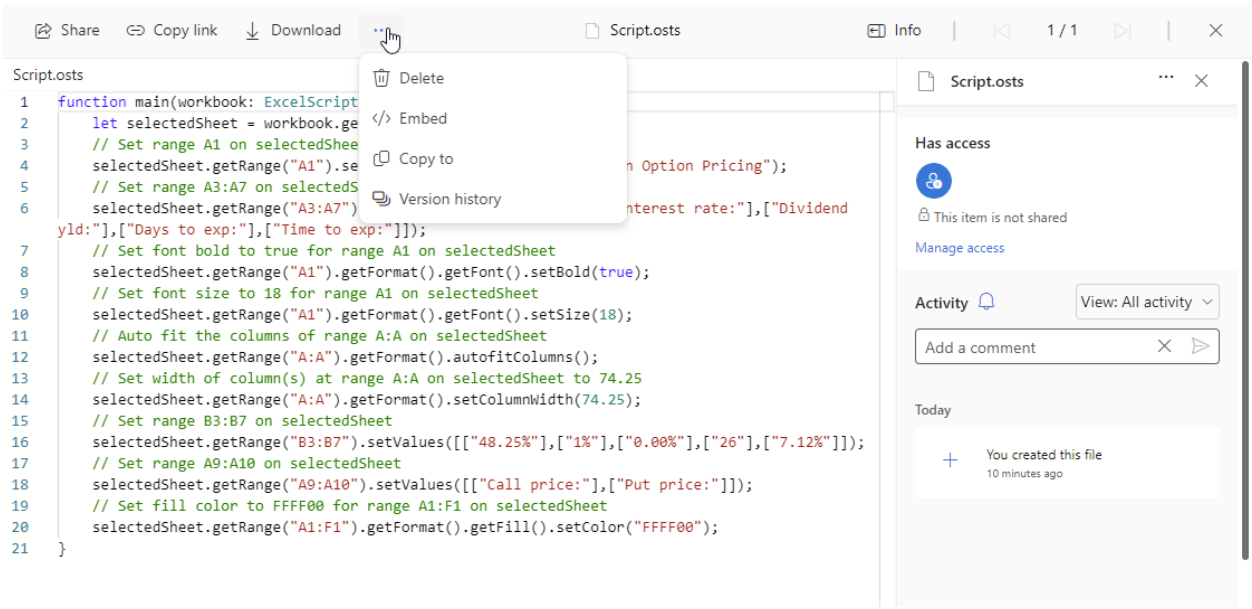
Nous choisissons un dossier et cliquons sur **Save**. Nous avons alors:



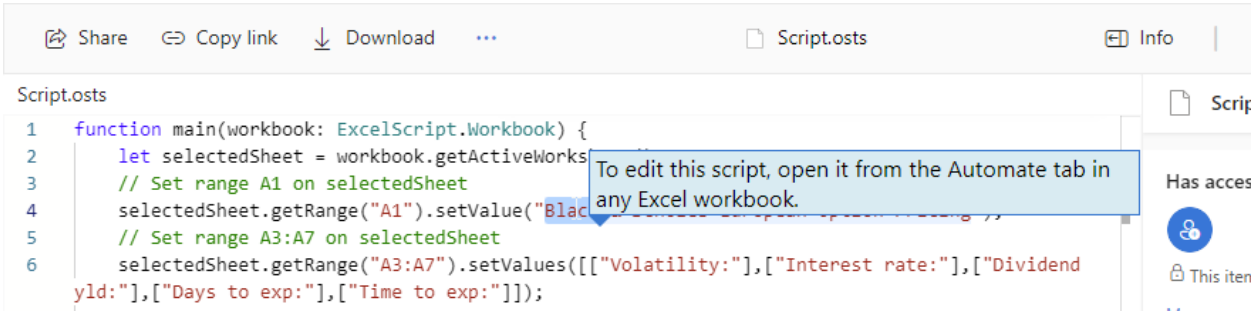
Il s'agit d'une bibliothèque SharePoint comme une autre comme l'atteste le menu contextuel du fichier:



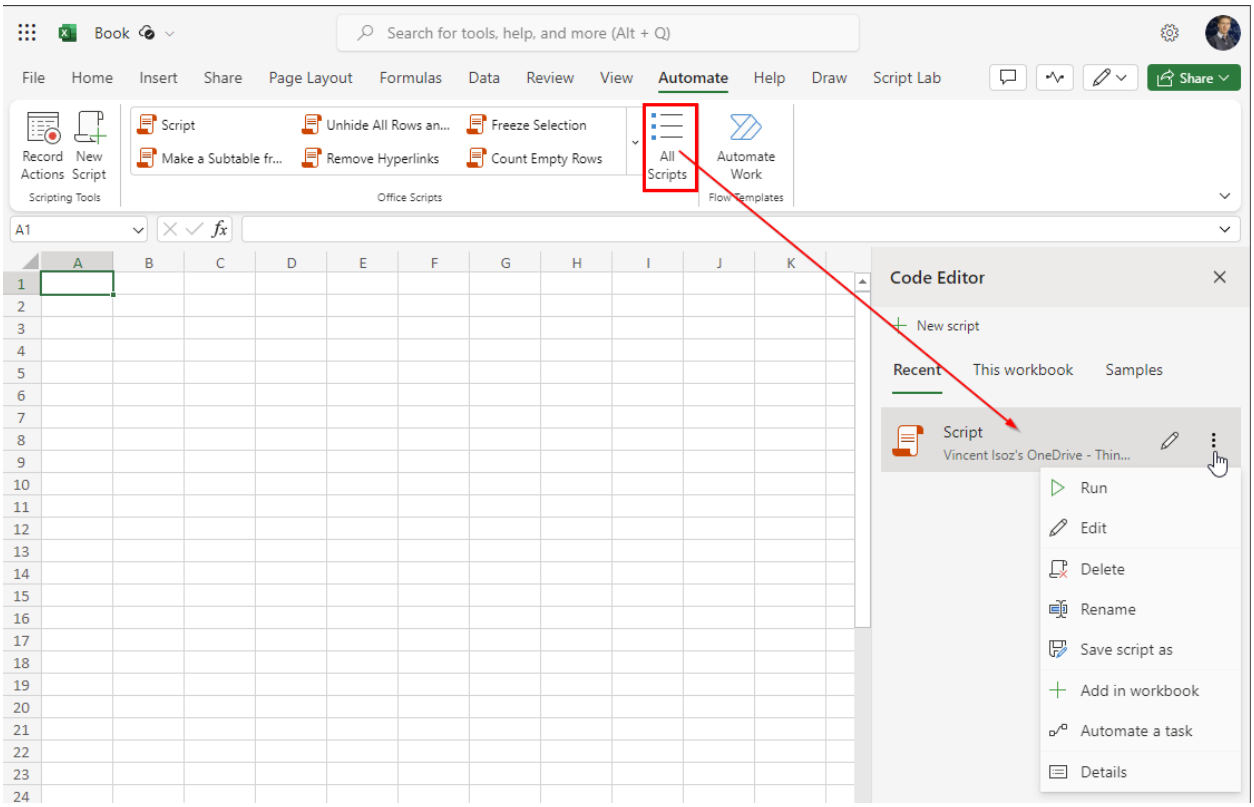
Ceci dit, si nous cliquons sur un fichier *.osts (Office Script Type Script):



Nous pouvons constater que les fichiers de script sont dans un éditeur en lecture seule malheureusement:

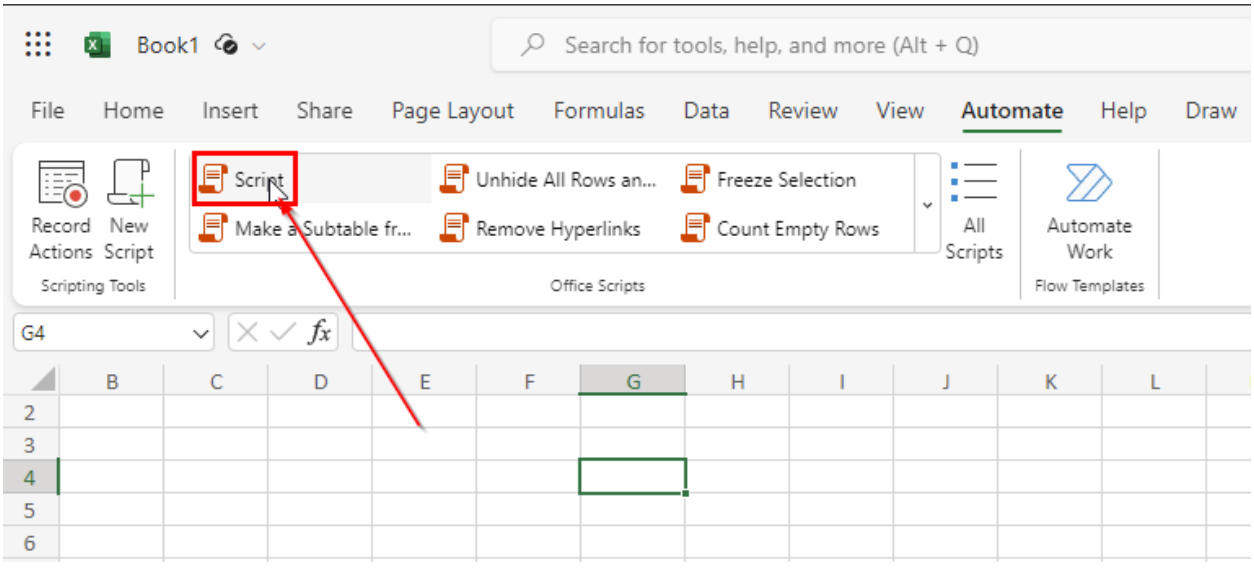


Nous pouvons aussi retrouver nos scripts enregistrés en allant dans **Automate All script**:



Ou simplement:

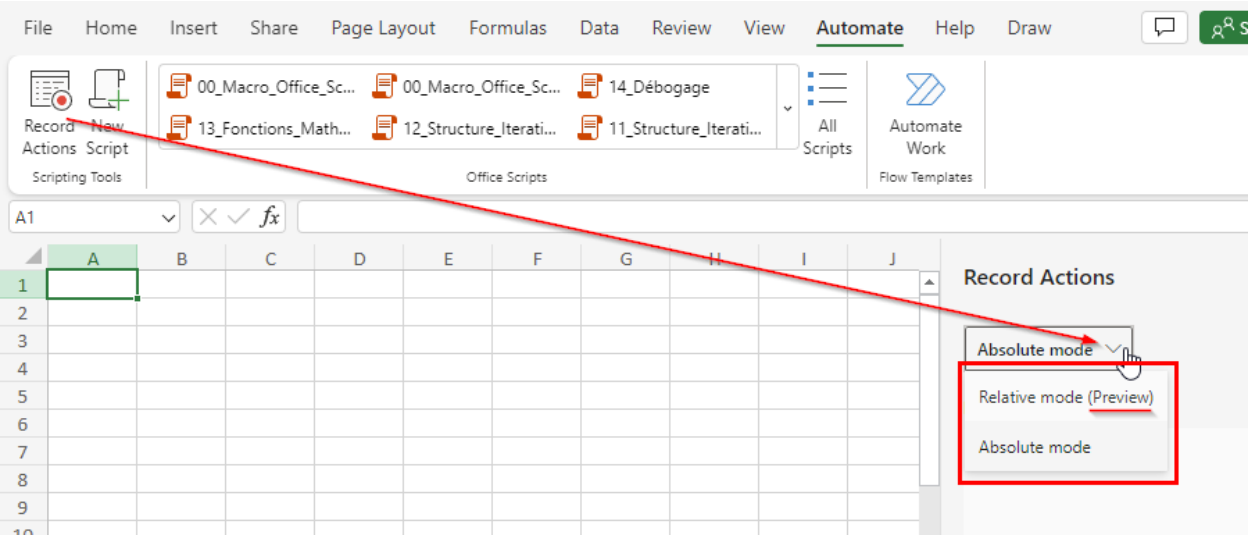
Public
Internal



Voilà! C'est tout pour l'enregistrement des script qui est très simple mais aussi comme en VBA très utile pour trouver la syntaxe de nombreuses commandes pour manipuler des graphiques, tableaux croisés dynamiques ou autres!

8.2 Macro relative et absolue

Au jour où nous écrivons ces lignes (8 décembre 2023), Microsoft a rajouté un bouton permettant de faire de l'enregistrement relatif ou absolu mais il ne semble pas fonctionner correctement après plusieurs tests (ce qui est normal vu qu'il est en ***Preview***):



Donc affaire à suivre…

Public
Internal

9 Écriture de scripts

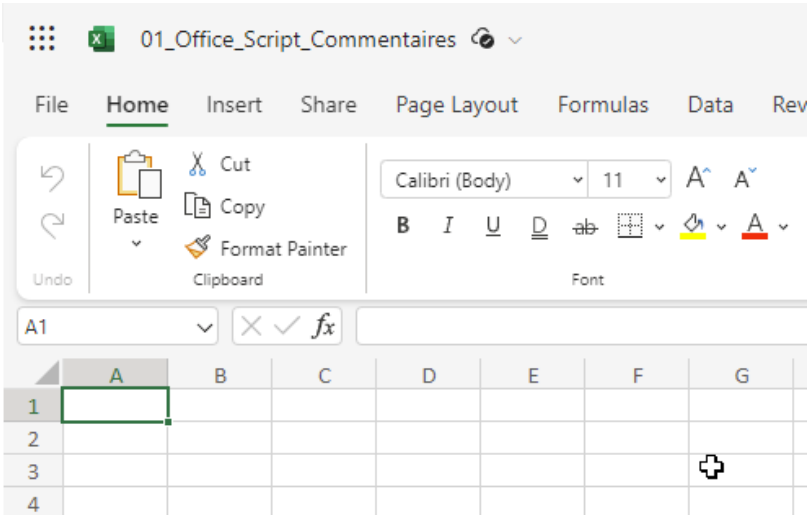
Nous allons commencer maintenant à écrire nos propres scripts basiques (nous sommes dans un cours sur les fondamentaux pour rappel!) depuis zéro en nous focalisant dans ce support sur les choses triviales que tout ingénieur apprend en première année de son cursus.

Commençons comme dans l'apprentissage de tout langage de script et de programmation par les commentaires…

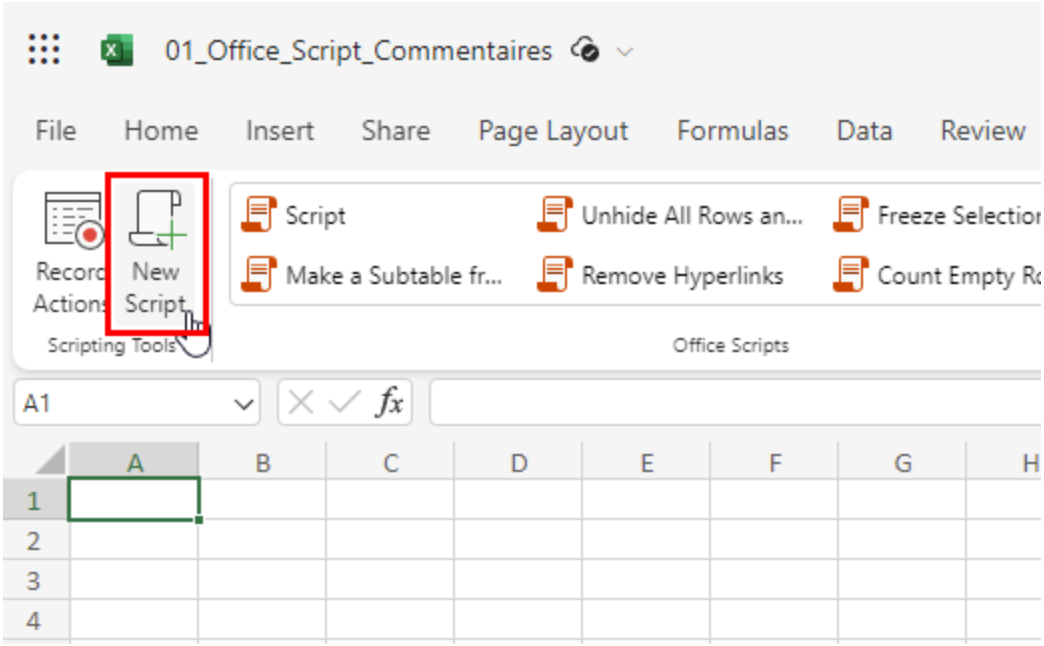
9.1 Commentaires

Pour voir comment créer des commentaires nous allons créer un nouveau fichier Excel et aussi un script…

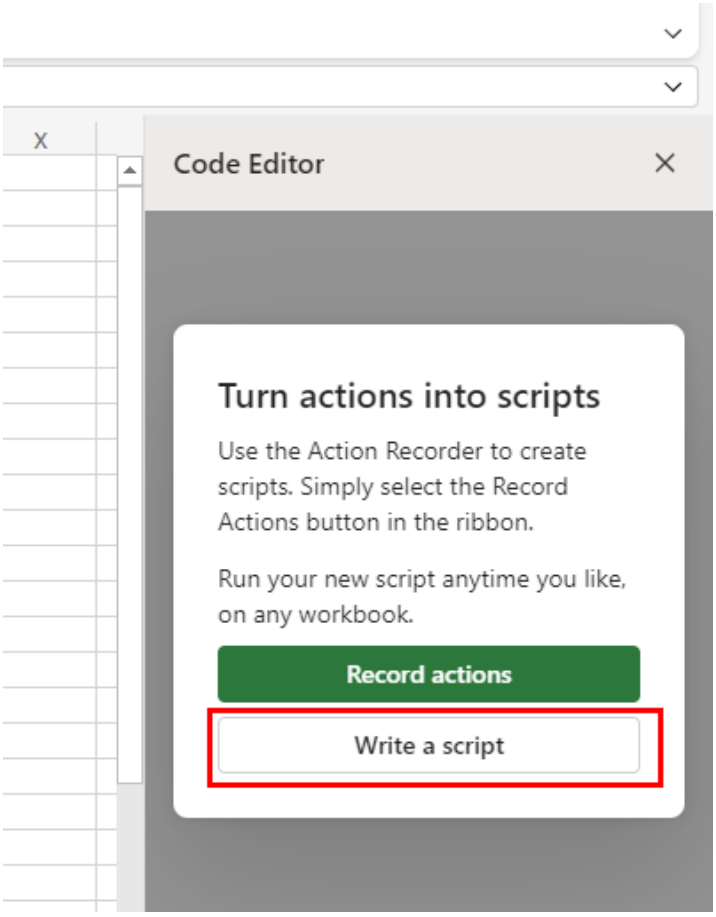
Créez le fichier suivant:



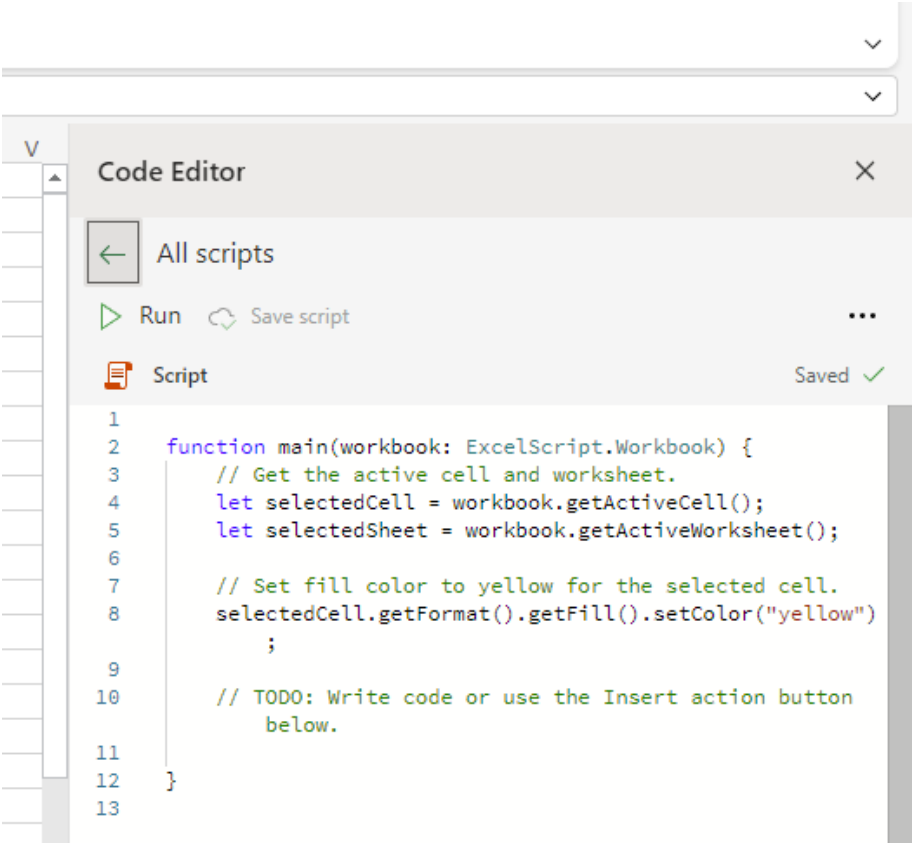
Cliquez sur **New Script**:



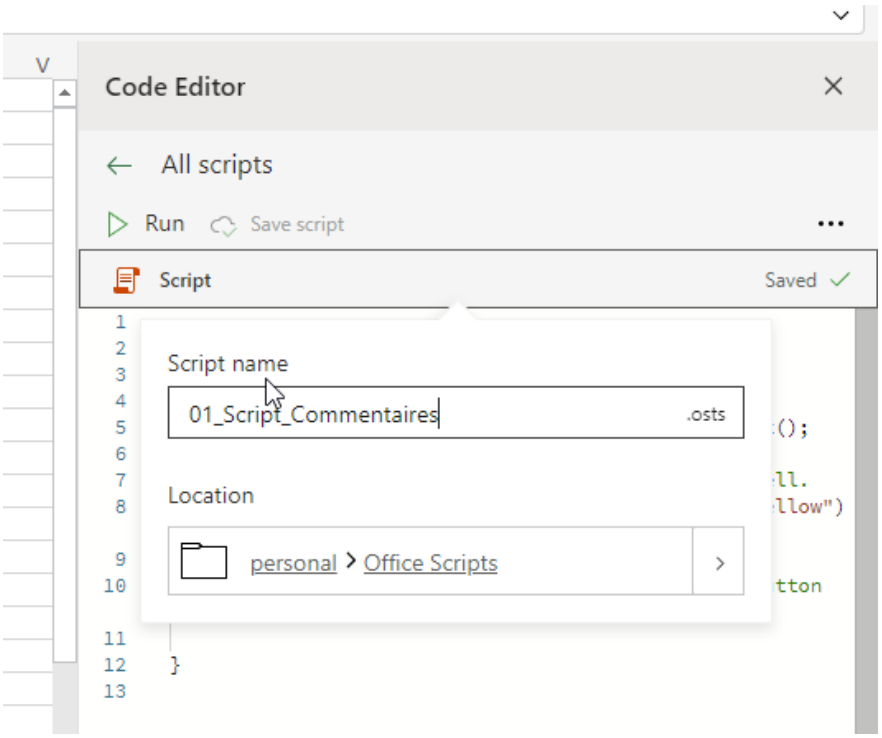
Nous confirmons que nous voulons créer un nouveau script écrit:



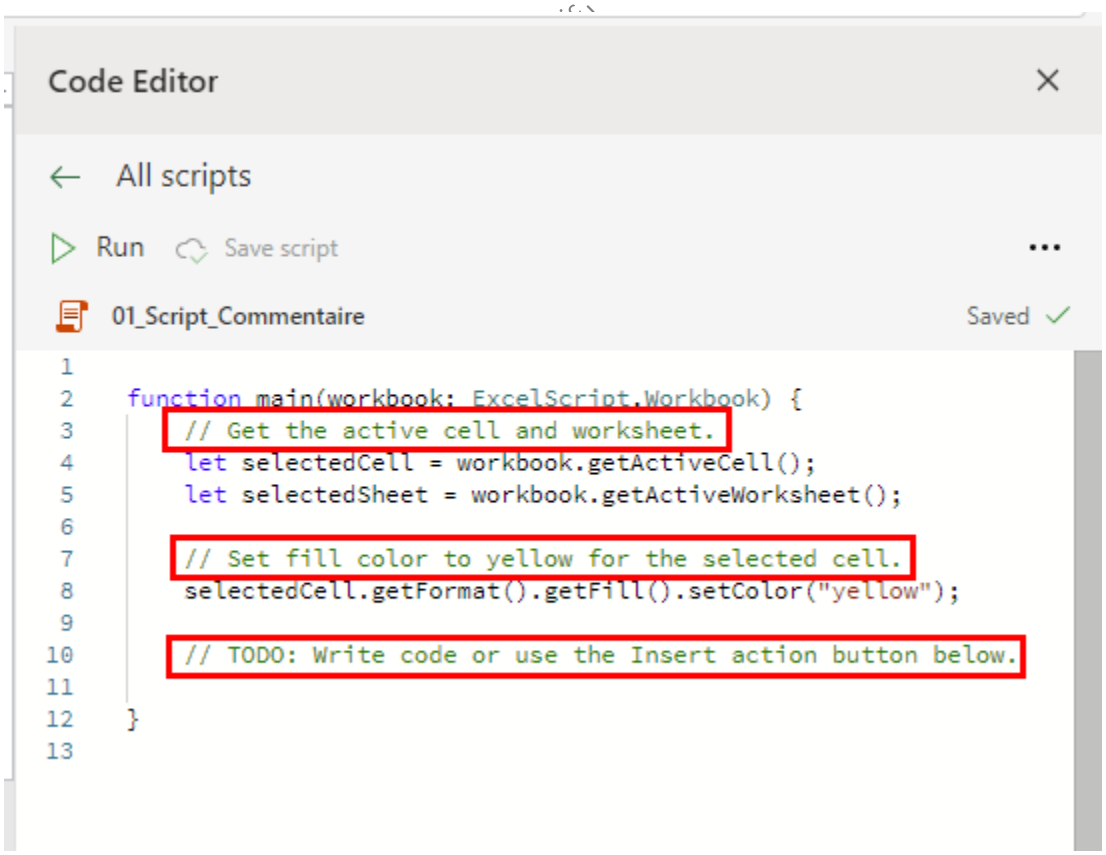
Une code écrit par défaut va alors apparaître:



Nous enregistrons ce script:



Et nous voyons déjà pour le coup le format type des commentaires en Office Script:

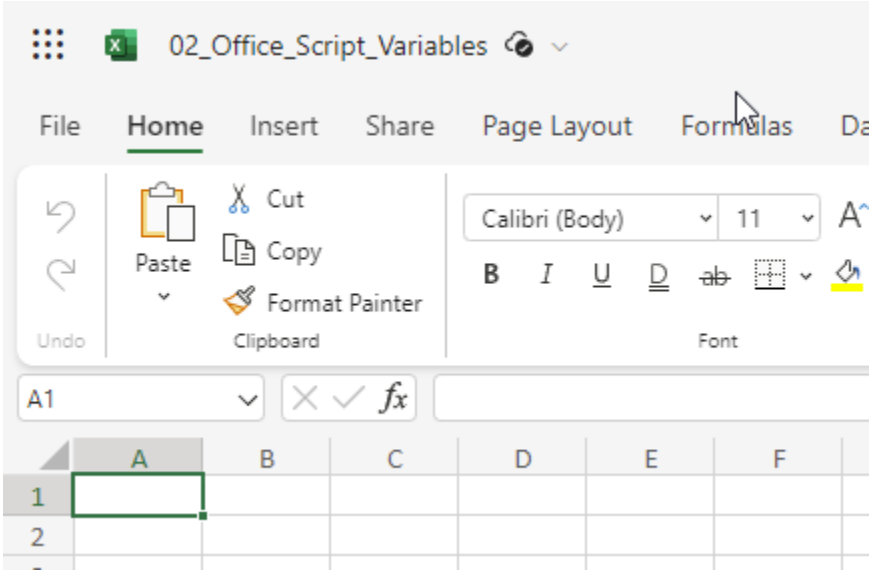


Ou pour écrire un commentaire sur plusieurs lignes:

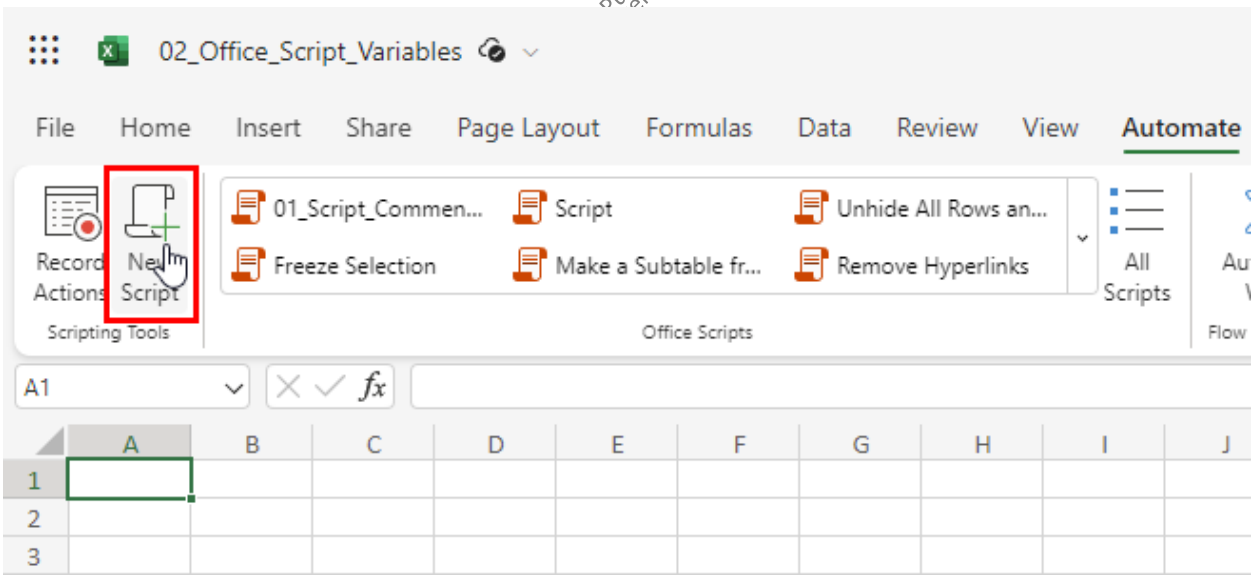
Et voilà! C'est tout pour les commentaires!

9.2 Variables et constantes

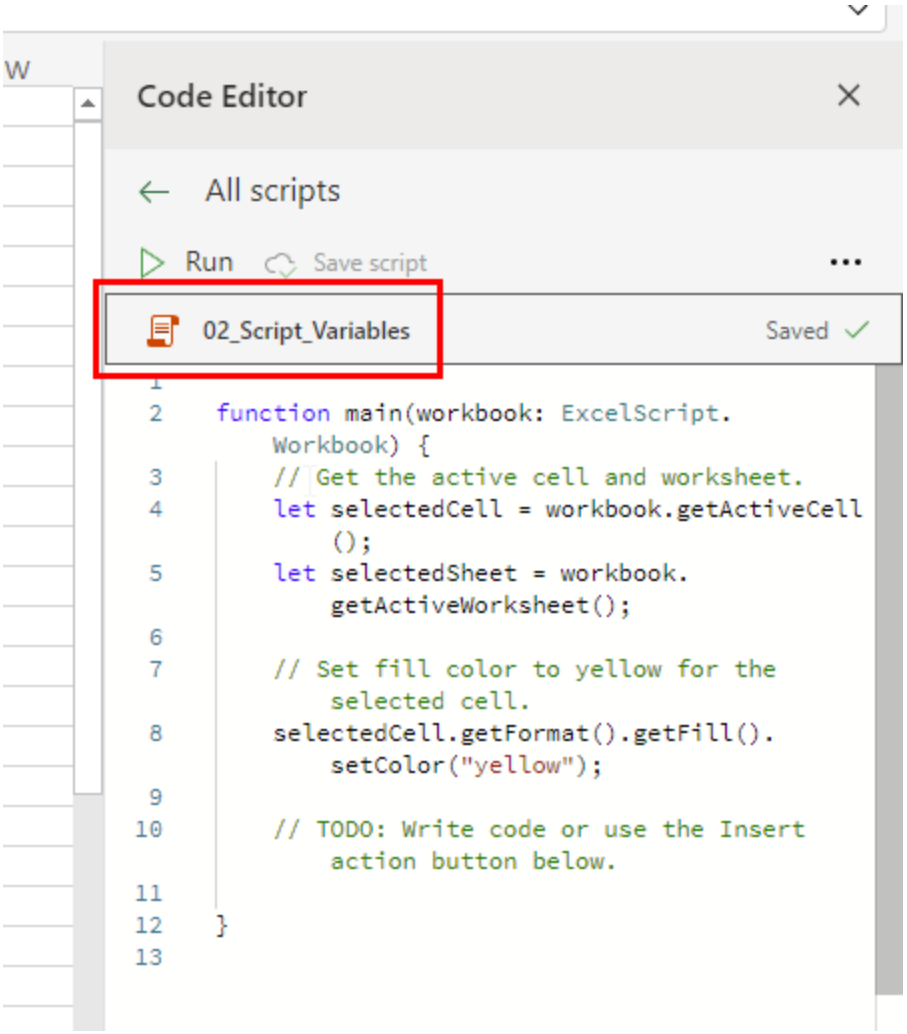
Voyons comment créer des variables et constantes en Office Script! Créez un nouveau fichier Microsoft Excel nommé *02_Office_Script_Variables.xlsx*:



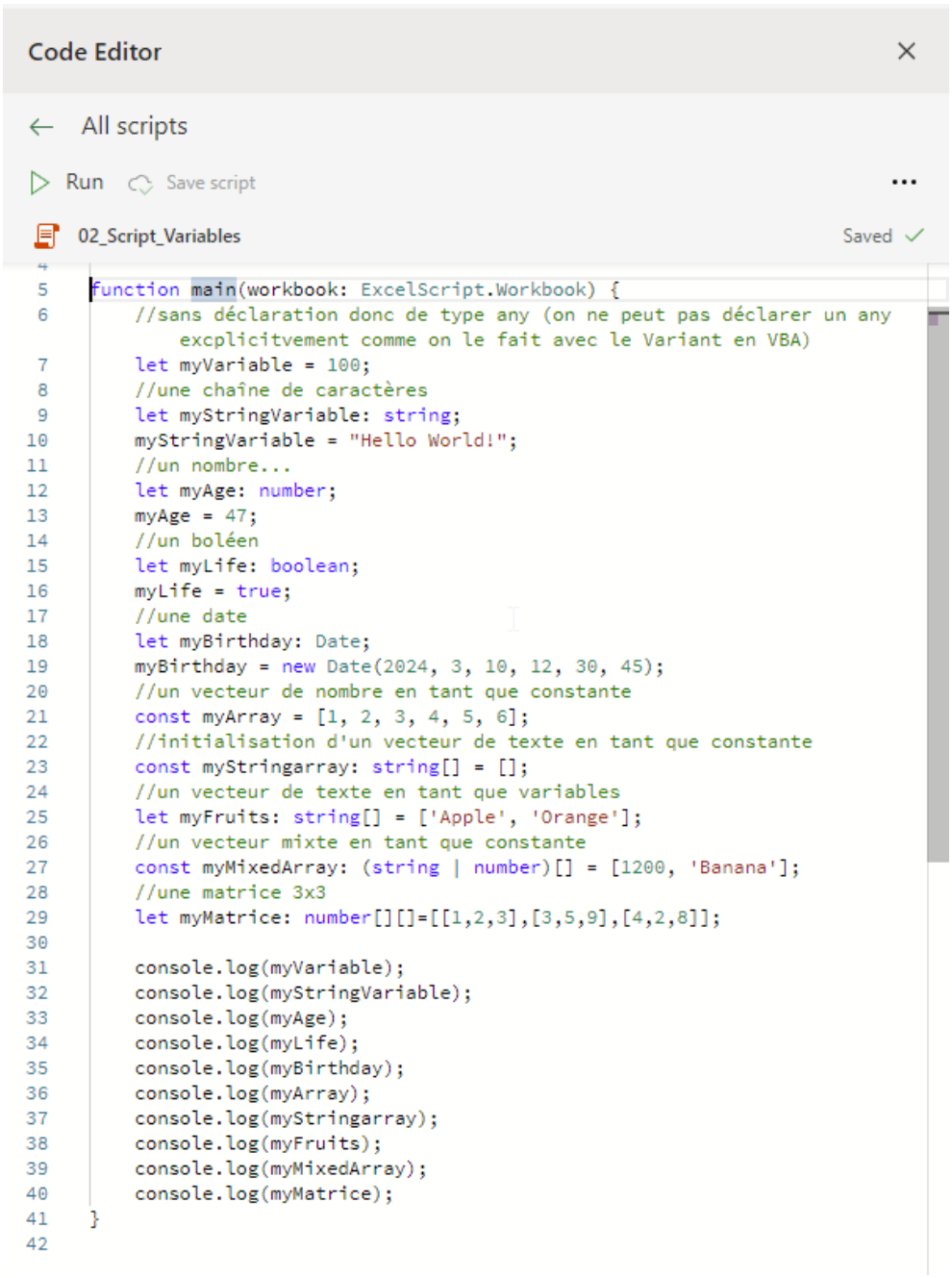
Ensuite, nous allons dans l'onglet **Automate** et cliquons sur **New Script**:



Et nous renommons le script crée **02_Script_Variables**:



Et créons le script suivant qui contient la grande majorité des ces les plus fréquents de déclaration de constantes et variables:

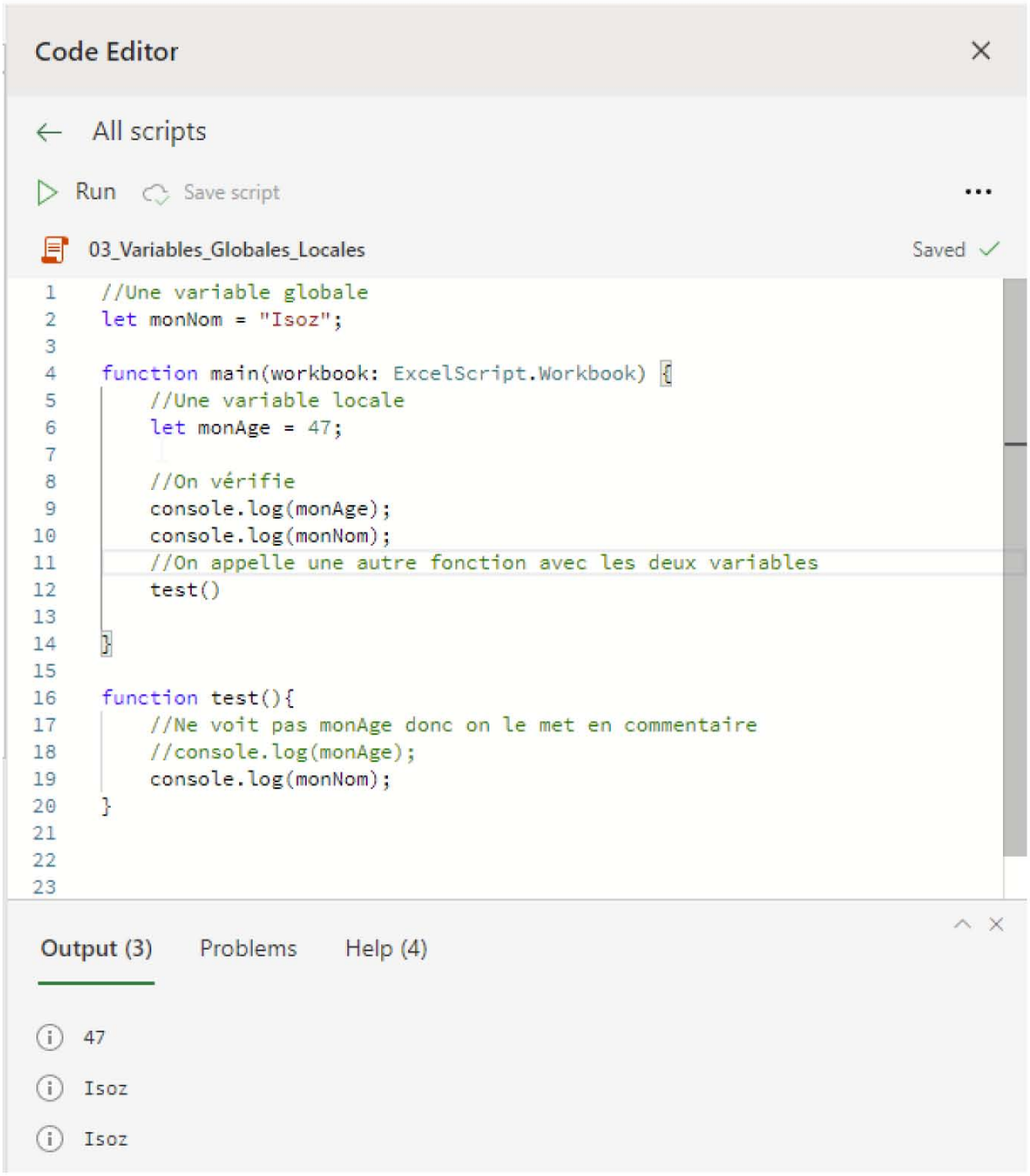



```typescript
function main(workbook: ExcelScript.Workbook) {
    //sans déclaration donc de type any (on ne peut pas déclarer un any excplicitvement comme on le fait avec le Variant en VBA)
    let myVariable = 100;
    //une chaîne de caractères
    let myStringVariable: string;
    myStringVariable = "Hello World!";
    //un nombre...
    let myAge: number;
    myAge = 47;
    //un boléen
    let myLife: boolean;
    myLife = true;
    //une date
    let myBirthday: Date;
    myBirthday = new Date(2024, 3, 10, 12, 30, 45);
    //un vecteur de nombre en tant que constante
    const myArray = [1, 2, 3, 4, 5, 6];
    //initialisation d'un vecteur de texte en tant que constante
    const myStringarray: string[] = [];
    //un vecteur de texte en tant que variables
    let myFruits: string[] = ['Apple', 'Orange'];
    //un vecteur mixte en tant que constante
    const myMixedArray: (string | number)[] = [1200, 'Banana'];
    //une matrice 3x3
    let myMatrice: number[][]=[[1,2,3],[3,5,9],[4,2,8]];

    console.log(myVariable);
    console.log(myStringVariable);
    console.log(myAge);
    console.log(myLife);
    console.log(myBirthday);
    console.log(myArray);
    console.log(myStringarray);
    console.log(myFruits);
    console.log(myMixedArray);
    console.log(myMatrice);
}
```

Les constantes ne sont pas disponibles sans initialisation. Les constantes ne peuvent évidemment pas être modifiées dans les programmes ultérieurs. Cependant, elles peuvent être affectées à d'autres variables comme en VBA (et autres langages).

9.2.1 Variables locales & globales

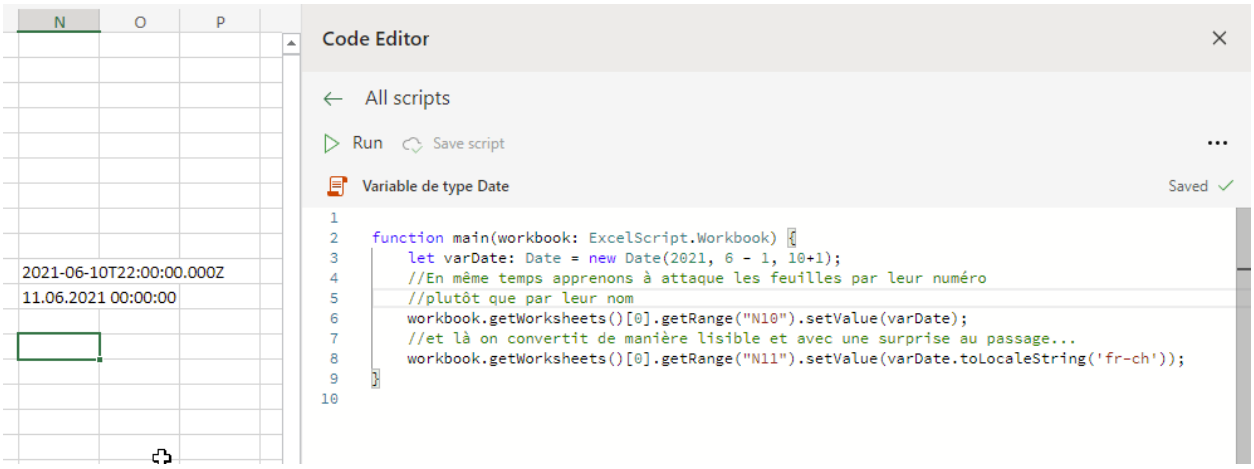
Un grand classique lorsqu'on écrit des programmes relativement long c'est de créer des variables à portée globale et locale. Voyons avec un exemple simple si cela existe en Office Script:



Nous voilà donc bien rassurés!

9.2.2 Spécificités des dates

Comme souvent la gestion des dates est un peu particulière. Nous avons déjà constaté cela plus haut lors de la déclaration des variables mais voyons que c'est bien évidemment le cas aussi lors de la lecture et écriture de dates dans des cellules.

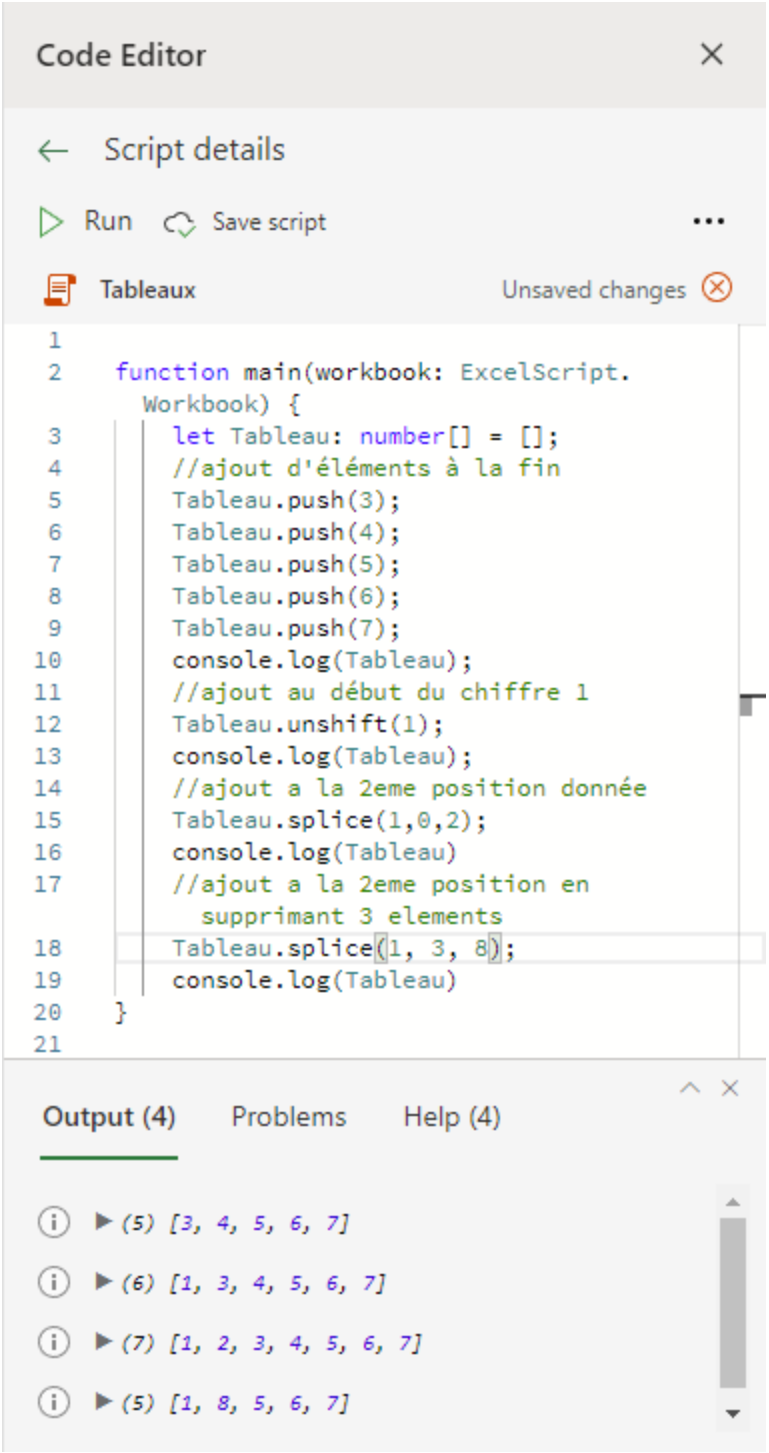


Donc que dire… un vrai bonheur comme d'habitude…

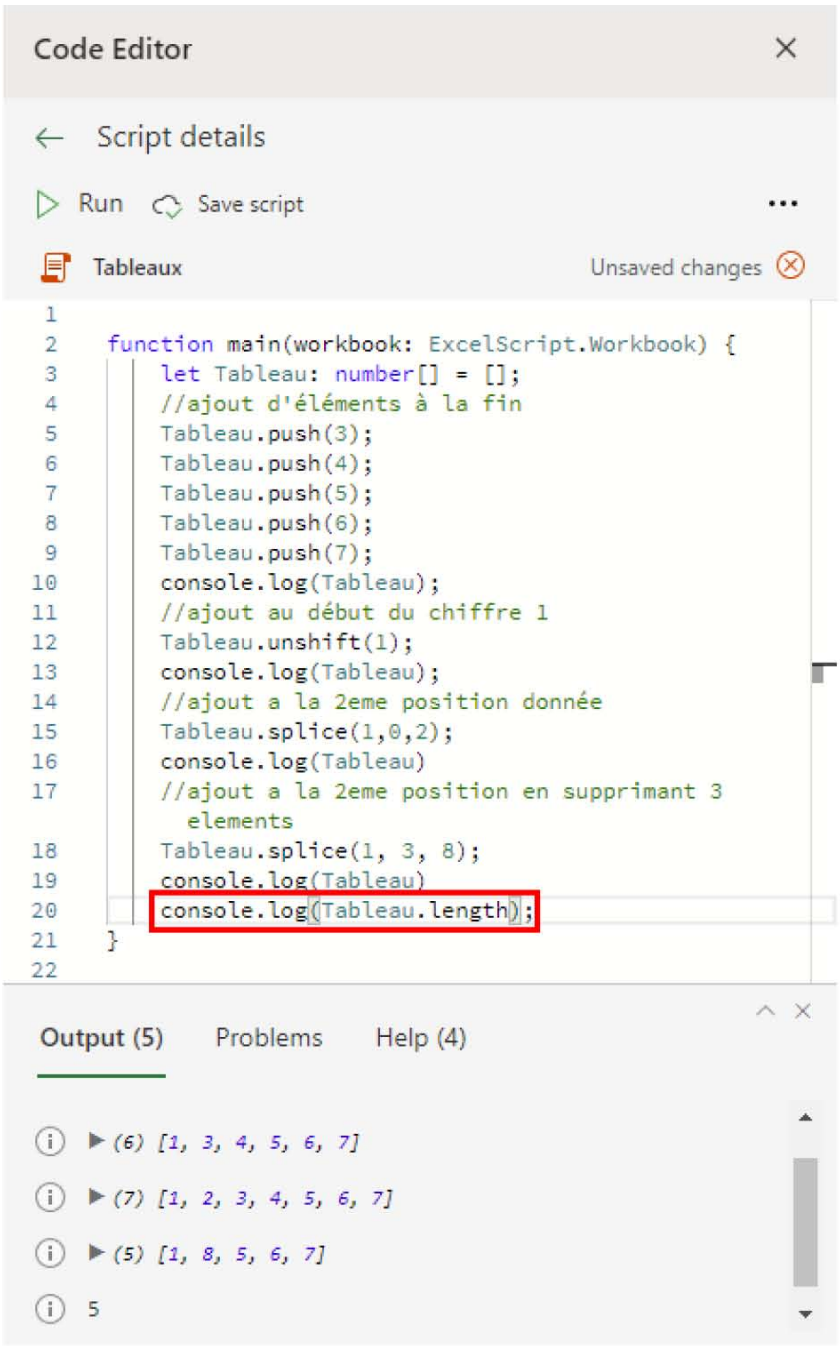
9.2.3 Spécificités des Tableaux (Arrays)

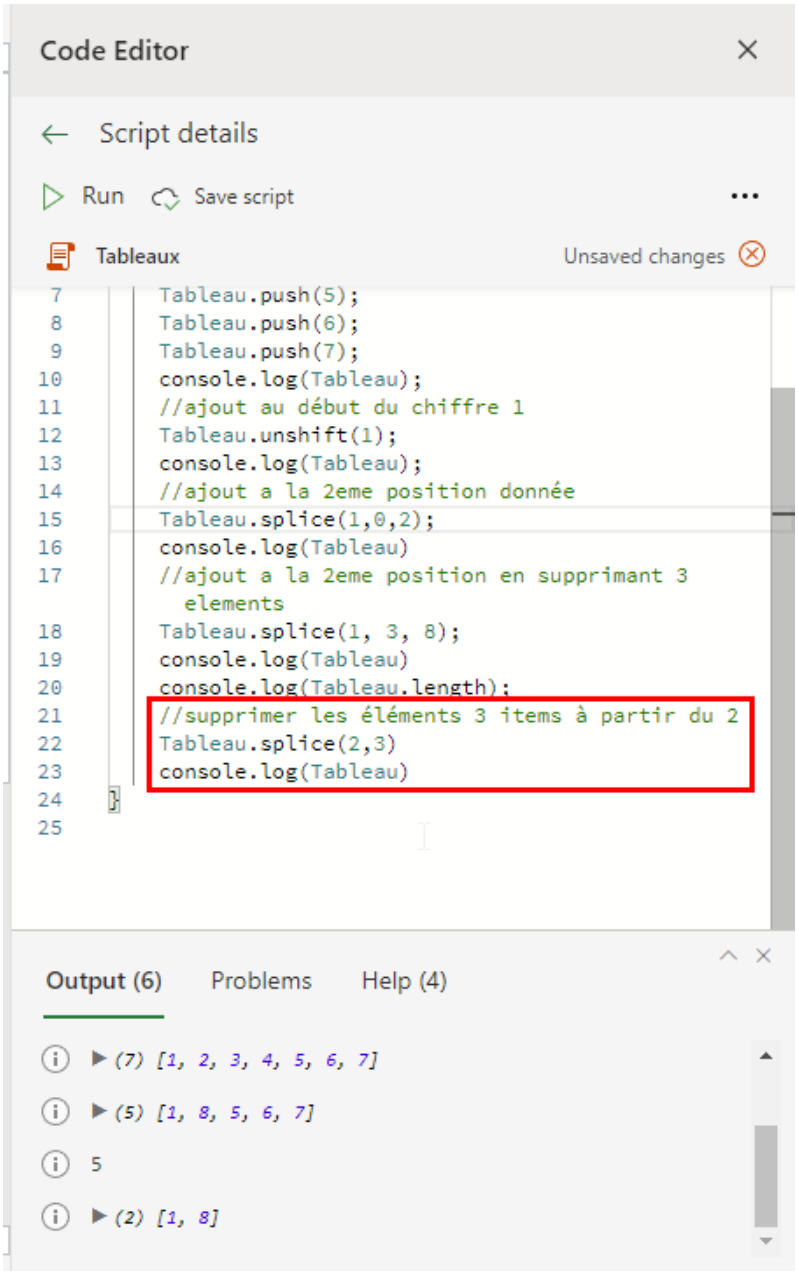
Public
Internal

Nous allons voir ici quelques éléments supplémentaires concernant les tableaux (arrays) comme l'ajout ou la substitution d'éléments:

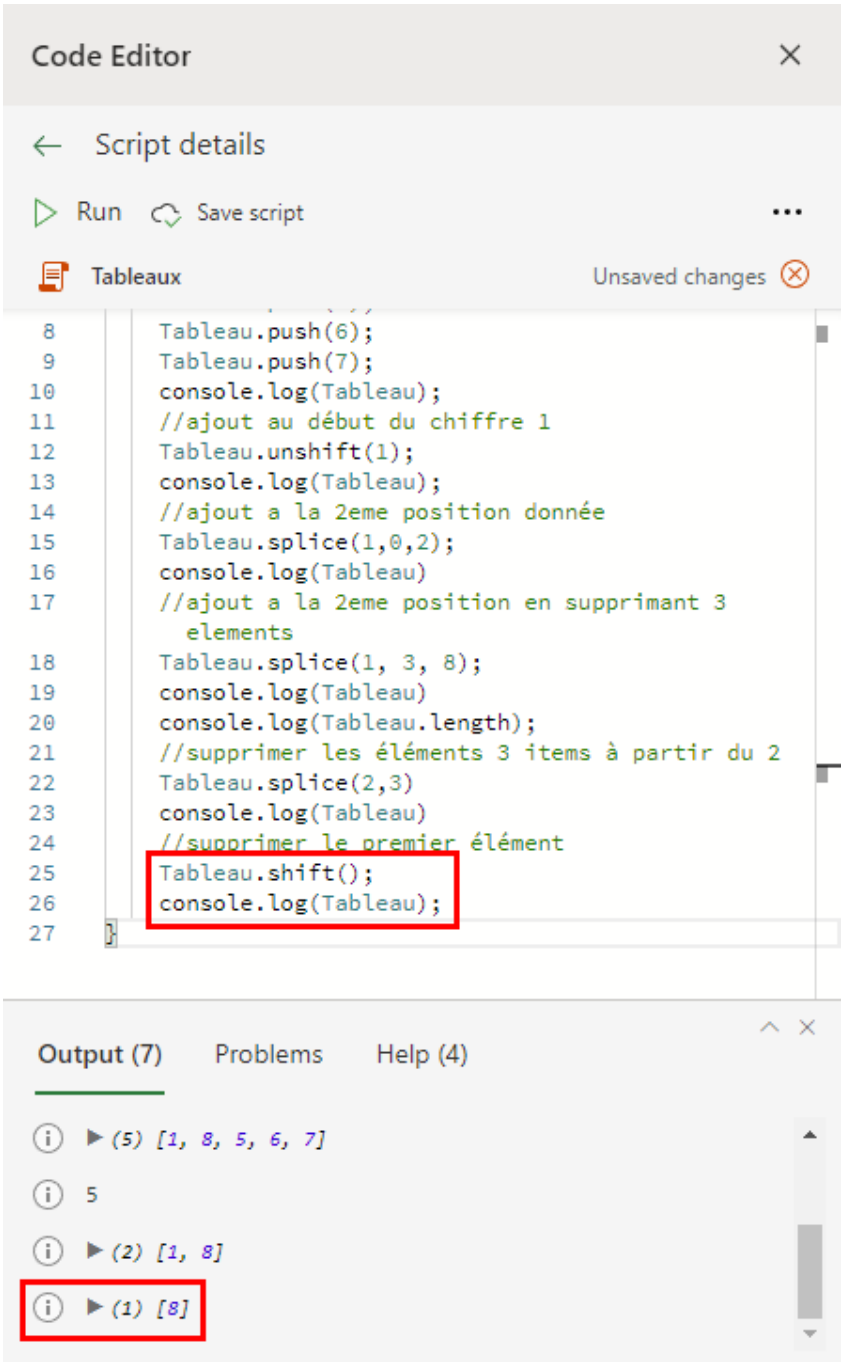


Continuons (faute de place) sur comment identifier la longueur d'un tableau:


Code Editor
Script details
Run
Save script
Tableaux
Unsaved changes
1
2 function main(workbook: ExcelScript.Workbook) {
3 let Tableau: number[] = [];
4 //ajout d'éléments à la fin
5 Tableau.push(3);
6 Tableau.push(4);
7 Tableau.push(5);
8 Tableau.push(6);
9 Tableau.push(7);
10 console.log(Tableau);
11 //ajout au début du chiffre 1
12 Tableau.unshift(1);
13 console.log(Tableau);
14 //ajout a la 2eme position donnée
15 Tableau.splice(1,0,2);
16 console.log(Tableau)
17 //ajout a la 2eme position en supprimant 3 elements
18 Tableau.splice(1, 3, 8);
19 console.log(Tableau)
20 console.log(Tableau.length);
21 }
22
Output (5)
Problems
Help (4)
▶ (6) [1, 3, 4, 5, 6, 7]
▶ (7) [1, 2, 3, 4, 5, 6, 7]
▶ (5) [1, 8, 5, 6, 7]
5



Ou (toujours faut de place) la suppression d'éléments en début de tableau:



Etc. Il y a plein d'autres choses que l'on peut faire avec les tableaux… et que j'écrirai le jour où je verrai des clients les utiliser comme:

- .concat (pour concaténer deux tableaux)
- .indexOf (pour trouver la première position d'un certain élément)
- .lastIndexOf (pour trouver la dernière position d'un certain élément)
- Math.max(null, Tableau) (pour trouver a valeur maximale d'un tableau)

- Math.min(null, Tableau) (pour trouver la valeur minimale d'un tableau)
- …

9.2.4 Créer des dictionnaires/hashes

À rédiger un jour pour le cours avancé s'il y a une demande ou que je le vois utilisé chez un client…

To Do

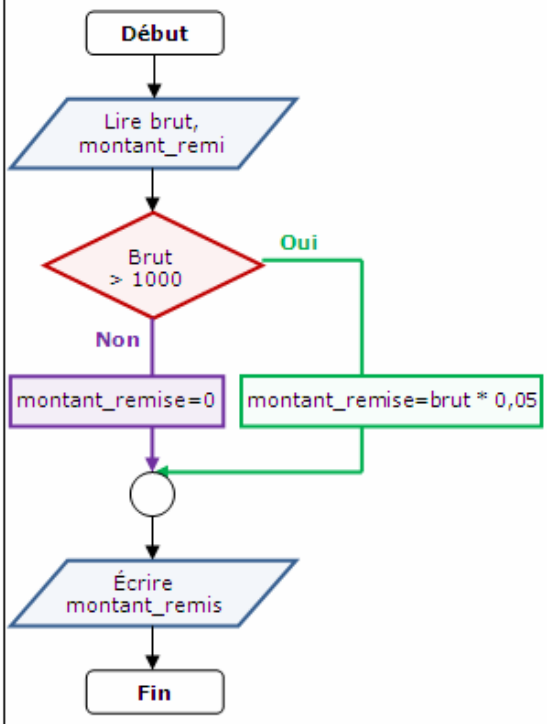
Public
Internal

9.3 Structures algorithmiques

Il existe deux types de structures conditionnelles en V.B.A. (comme dans la majorité des langages de programmation). On peut toujours utiliser l'un ou l'autre à ma connaissance mais les développeurs ont des préférences d'écriture qui s'imposent d'elles-mêmes en termes de facilité de relecture du code.

Calcul d'une remise en fonction du brut d'une facture

Organigramme	Pseudo-code

	LIRE brut, montant_remise
	DEBUT
	  SI brut > 1000
	  ALORS montant_remise = brut * 0,05
	  SINON montant_remise = 0
	  FINSI
	ECRIRE montant_remise
	FIN

Ces deux types de structures conditionnelles sont IF et SELECT. En général dans les applications financières et d'ingénierie demandant une grande rapidité on préféra le SELECT qui est statistiquement plus rapide que IF de 5% à 20% dans la majorité des cas.

9.3.1 Structure Conditionnelle

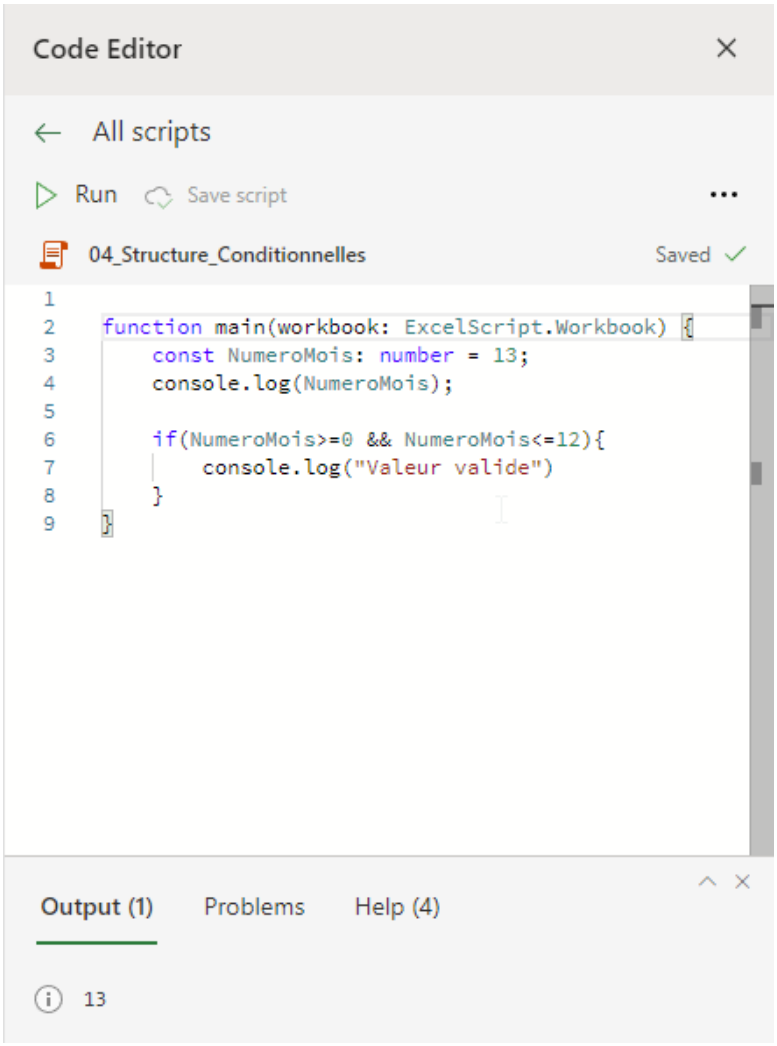
D'abord un rappel des opérateurs de comparaison en Office Script:

Opérateur	Contenu	Exemple
<	plus petit que	Valeur A est plus petit que valeur B
<=	plus petit ou égal à	Valeur A est plus petite ou égale à B
>	plus grand que	Valeur A est plus grand que valeur B
>=	plus grand ou égal à	Valeur A est plus grand ou égale à B
= =	équivalent	Valeur A égale valeur B

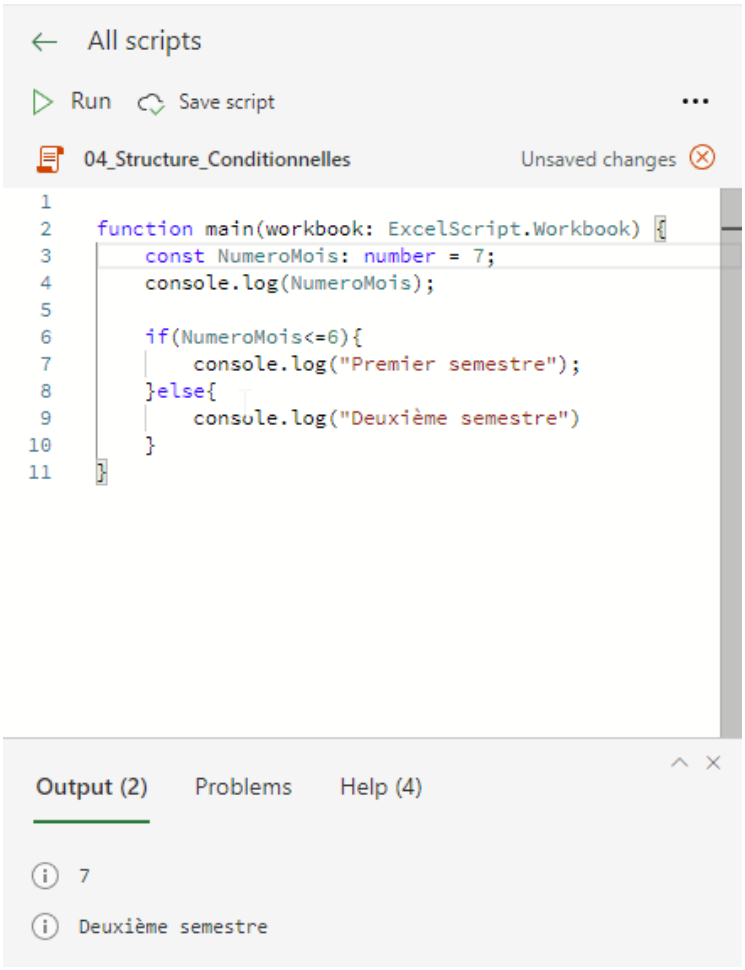
!=	différent de	Valeur A différente de valeur B
&&	Opérateur ET (conjonction logique)	Conditions A et B sont toutes deux vraies
!!	Opérateur OU	Conditions A ou B sont vraies

9.3.1.1 Structures IFs

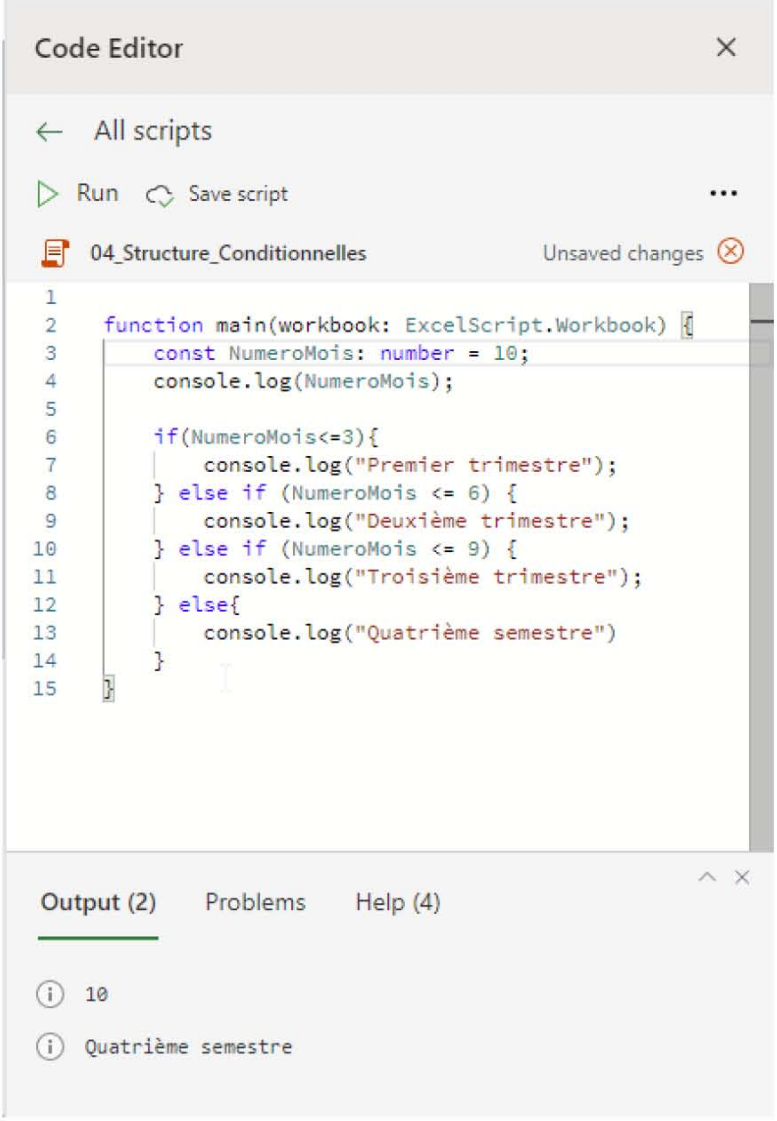
Voyons d'abord un structure Si classique ("vanille"):



Et ensuite une structure Si… Sinon classique ("vanille"):



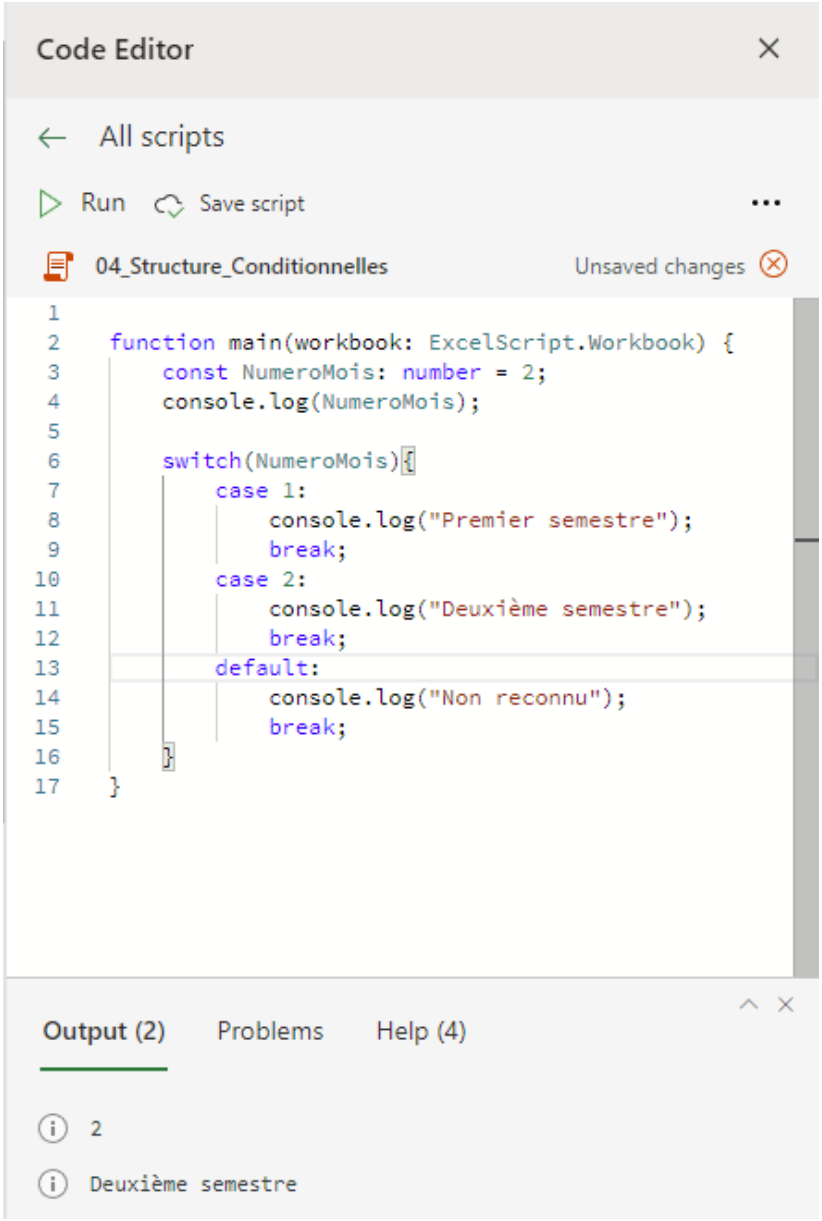
Et enfin la structure classique Si, alors… sinon, sinon, sinon… :


Code Editor
All scripts
Run
Save script
04_Structure_Conditionnelles
Unsaved changes
1
2 function main(workbook: ExcelScript.Workbook) {
3 const NumeroMois: number = 10;
4 console.log(NumeroMois);
5
6 if(NumeroMois<=3){
7 console.log("Premier trimestre");
8 } else if (NumeroMois <= 6) {
9 console.log("Deuxième trimestre");
10 } else if (NumeroMois <= 9) {
11 console.log("Troisième trimestre");
12 } else{
13 console.log("Quatrième semestre")
14 }
15 }
Output (2)
Problems
Help (4)
10
Quatrième semestre

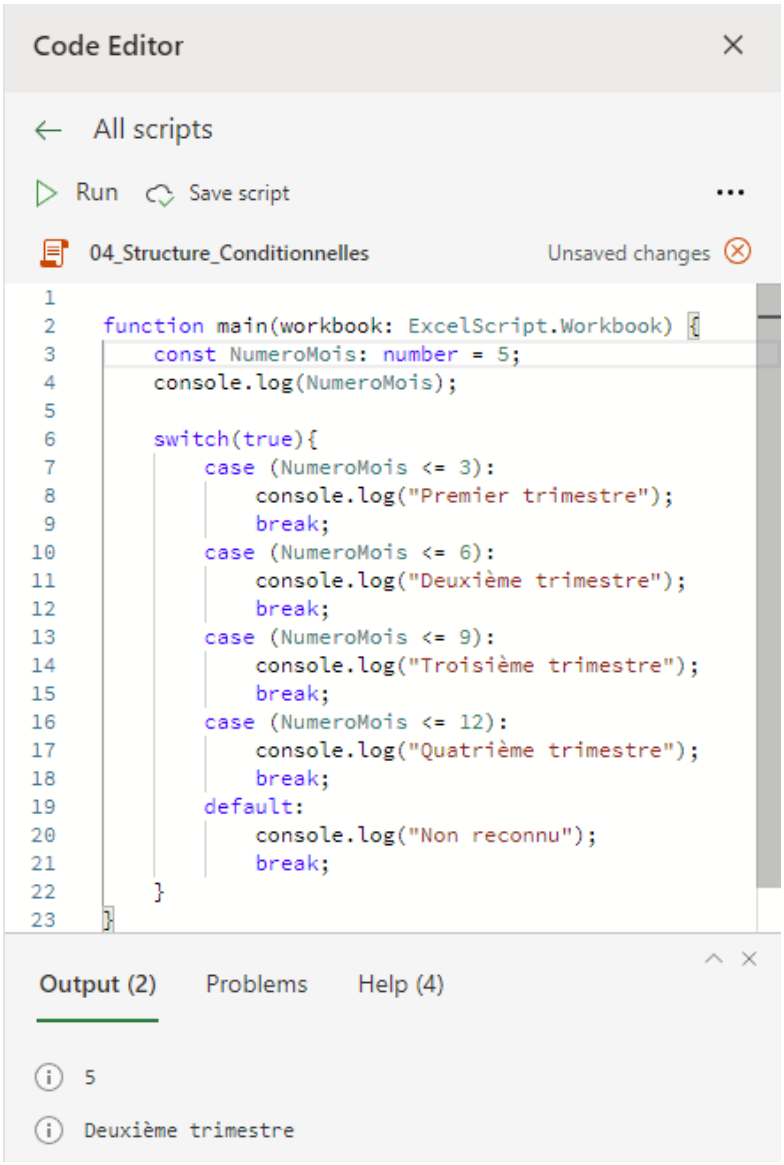
9.3.1.2 Structures Switchs

Les switch comme nous le savons permettent de simplifier la lecture dei Si multiples et aussi parfois d'accélérer le traitement des données en comparaisons à de multiples Si…. Alors… Sinon.

Voyons un exemple simple avec égalités strictes:



Et lorsque nous avons de nombreuses conditions avec autre chose qu'une égalité stricte, la syntaxe est:

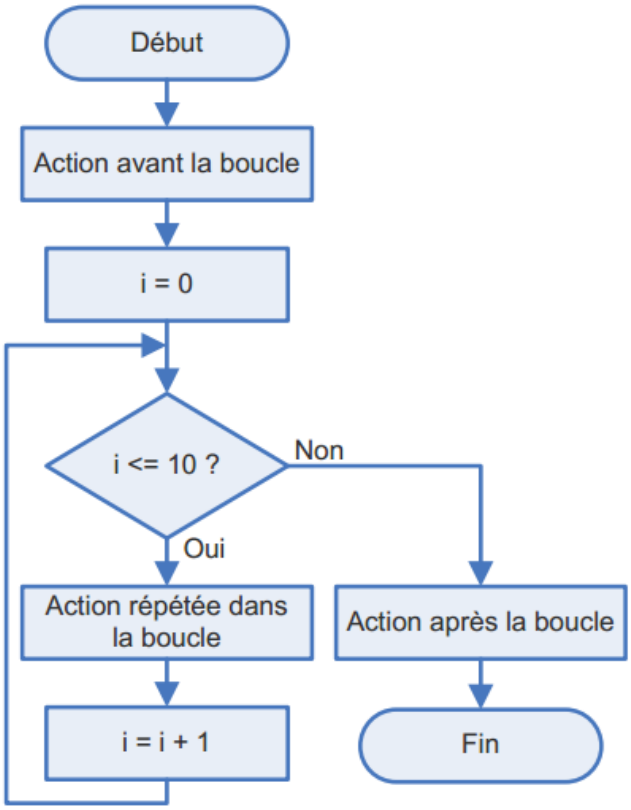


9.3.2 Boucles

Rappelons d'abord les trois types de boucles les plus courantes:

Nom	Caractéristique
for	répétition par un compteur
while	répétition tant que certaines conditions ne sont pas rencontrées en bout de boucle
do-while	répétition tant que certaines conditions ne sont pas rencontrées en début de boucle

Et sous forme de flux pour rappel voici le concept d'une boucle type pour ceux qui auraient oublié ou n'auraient pas suivi le cours d'algorithmique au préalable:

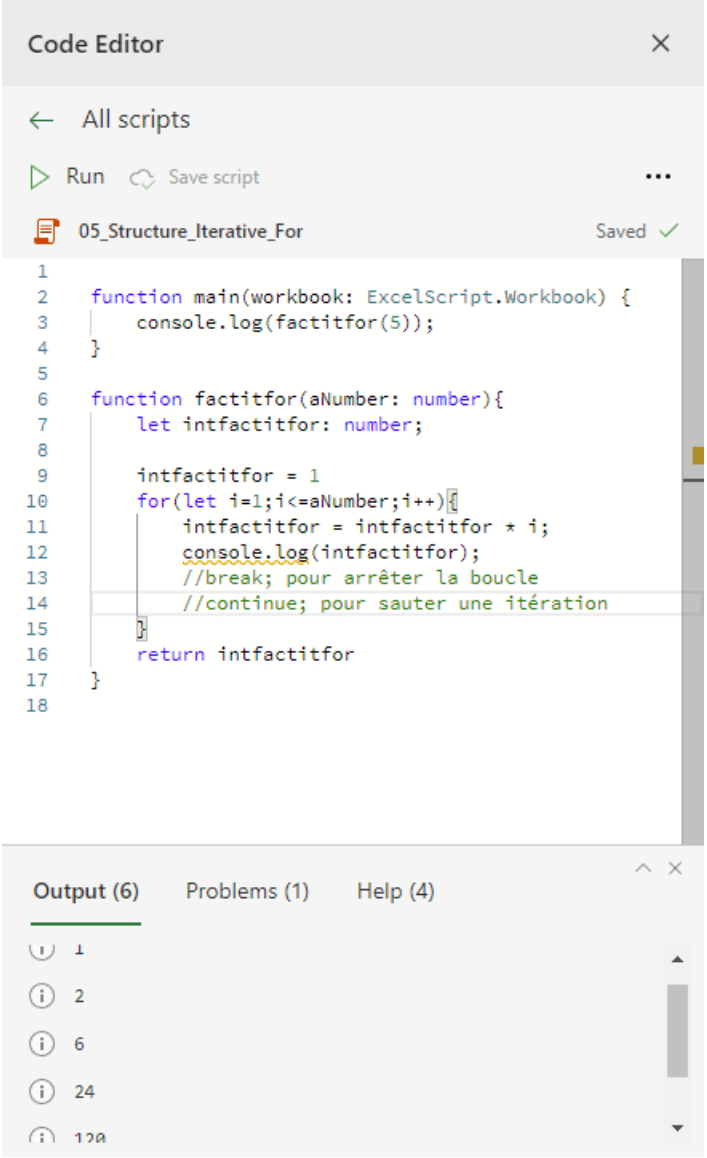


Public
External

Voyons un exemple de chaque avec le même cas exemple que dans le cours VBA, R, Python, C++.

9.3.2.1 For loop

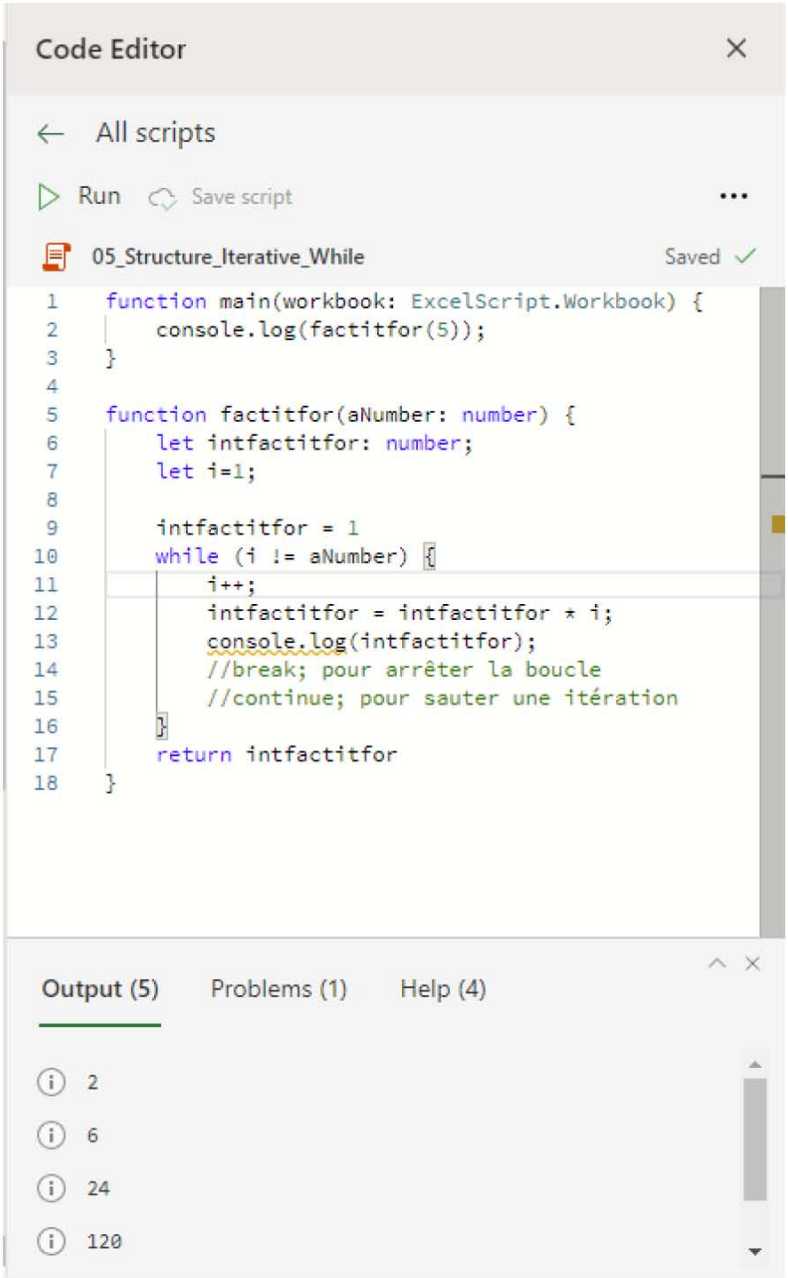
Nous commençons donc avec la boucle la plus intuitive qui est la *For…*:



Nous verrons évidemment des boucles plus utiles que ce cas abstrait et académique plus tard!

9.3.2.2 While

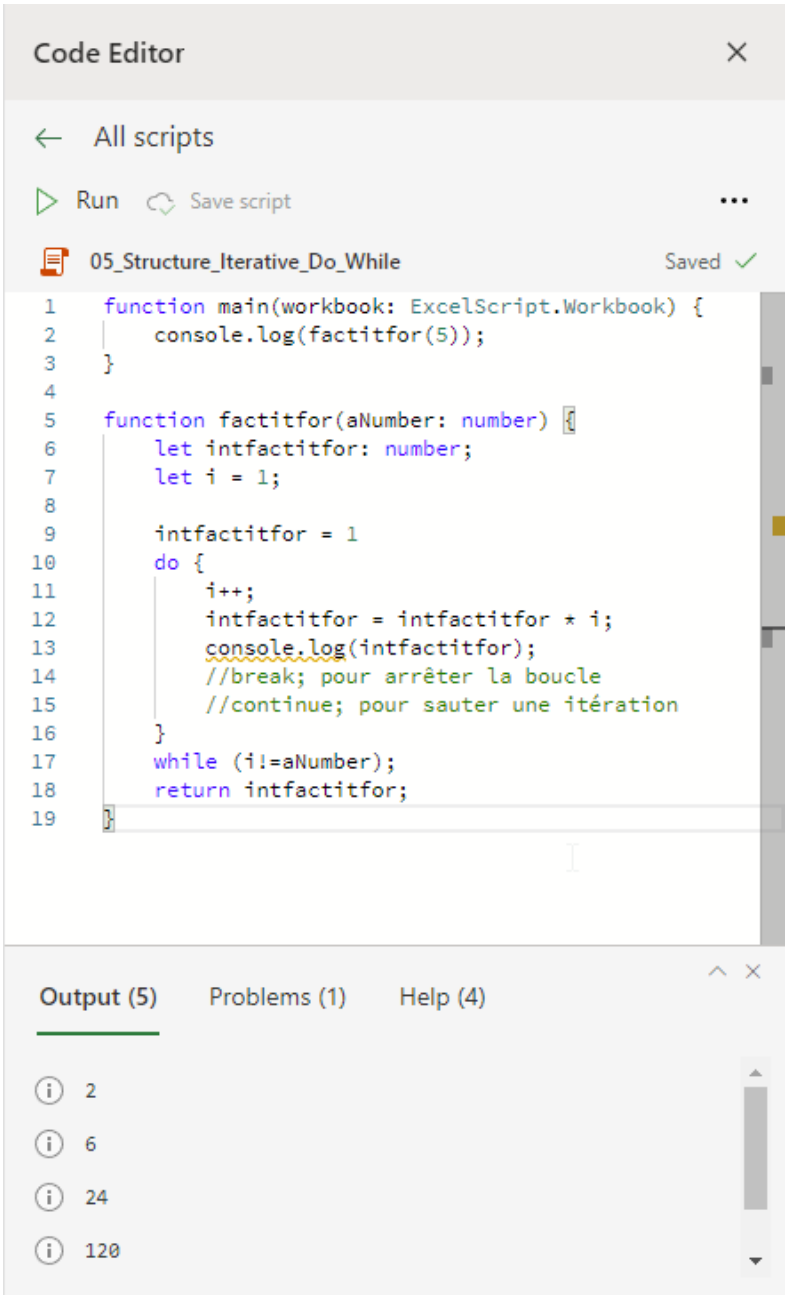
Passons à la deuxième boucle la plus courante qu'est le boucle *While* avec le même cas que l'exemple précédent:



Nous verrons évidemment des boucles plus utiles que ce cas abstrait et académique plus tard!

9.3.2.3 Do-While

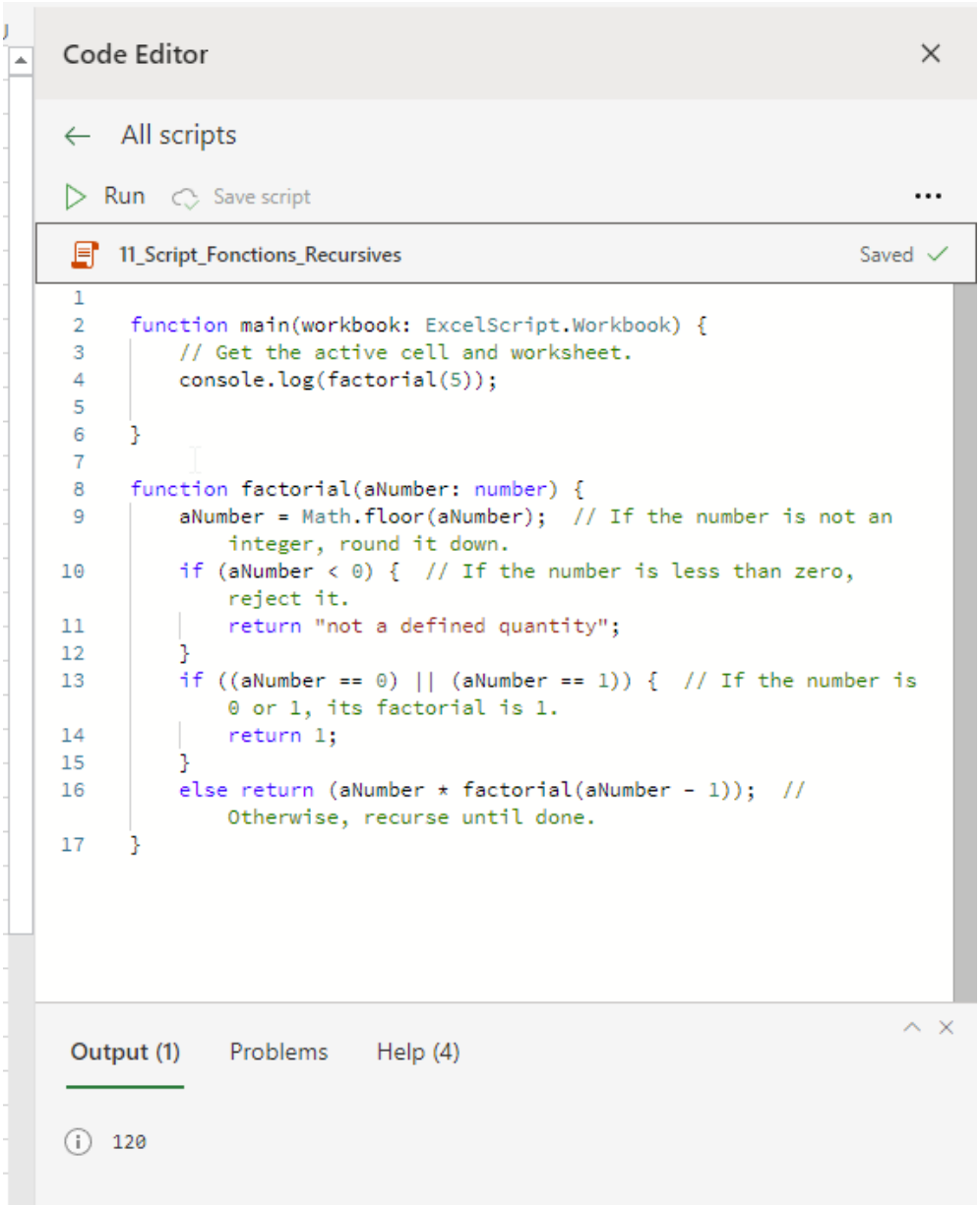
Et enfin le boucle la plus rarement rencontrée dans la pratique qui est la *Do… While*:



Nous verrons évidemment des boucles plus utiles que ce cas abstrait et académique plus tard!

9.3.3 Fonctions récursive (passage de paramètres)

Voyons maintenant le fameux cas des boucles récursives très connues dans les mathématiques appliquées de relativement haut niveau (comme par exemple le calcul du déterminant d'une matrice par les méthode des co-facteurs), mais ici avec un cas simple consistant à calculer la factorielle d'un nombre entier:



9.4 Fonctions mathématiques

Les fonctions de classeur VBA ne sont pas disponibles à ce jour en Office Script. Cependant elles sont disponibles en Office.js comme nous l'avons déjà mentionné et le lecteur pourra le vérifier ici:

https://learn.microsoft.com/en-us/office/dev/add-ins/excel/excel-add-ins-worksheet-functions

Les seules fonctions nativement disponibles à ce jour en Office Script sont données dans la liste complète des fonctions mathématiques ici:

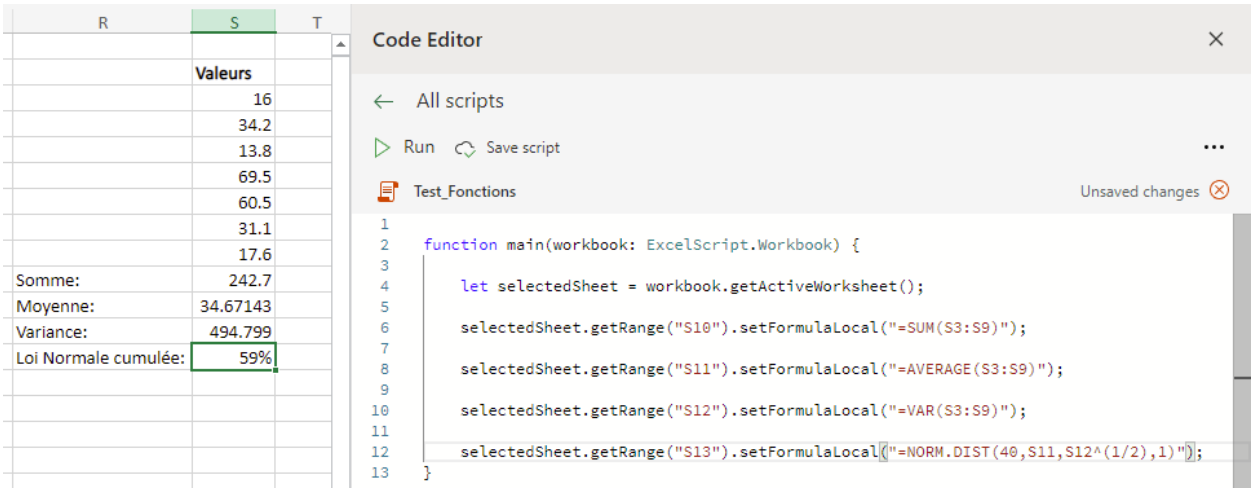
https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Math

Voici un petit échantillon de cette page résumé sous forme tabulaire (les constantes sont rouge):

Math.E	Math.LN10	Math.LN2	Math.LOG10E	Math.LOG2E
Math.PI	Math.SQRT1_2	Math.SQRT2	Maths.abs( )	Math.acos( )
Math.acosh( )	Math.asin( )	Math.asinh( )	Math.atan( )	Math.atan2( )
Math.atanh( )	Math.cbrt( )	Math.ceil( )	Math.cos( )	Math.cosh( )
Math.exp( )	Math.floor( )	Math.fround( )	Math.hypot( )	Math.imul( )
Math.log( )	Math.log10( )	Math.max( )	Math.min( )	Math.pow( )
Math.random( )	Math.round( )	Math.sign( )	Math.sin( )	Math.sinh( )
Math.sqrt( )	Math.tan( )	Math.tanh( )	Math.trunc( )	…

Propriété intellectuelle

Voyons cependant avec une bête macro enregistrée comme Office Script enregistre les fonctions mathématiques de Excel:

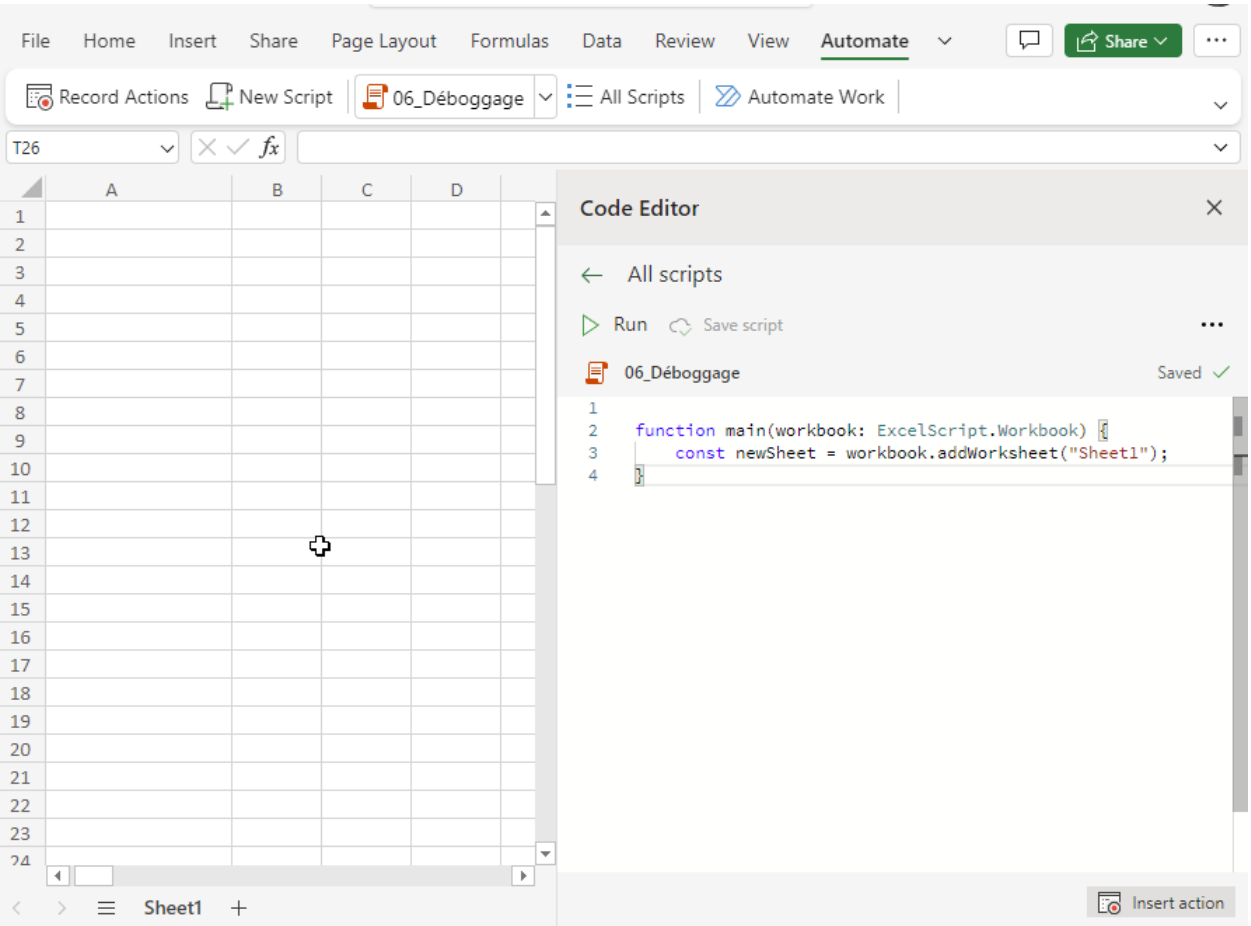


Cela nous montre que l'enregistreur de macros est capable d'enregistrer correctement l'écriture de toutes les fonctions Microsoft Excel dans les cellules!

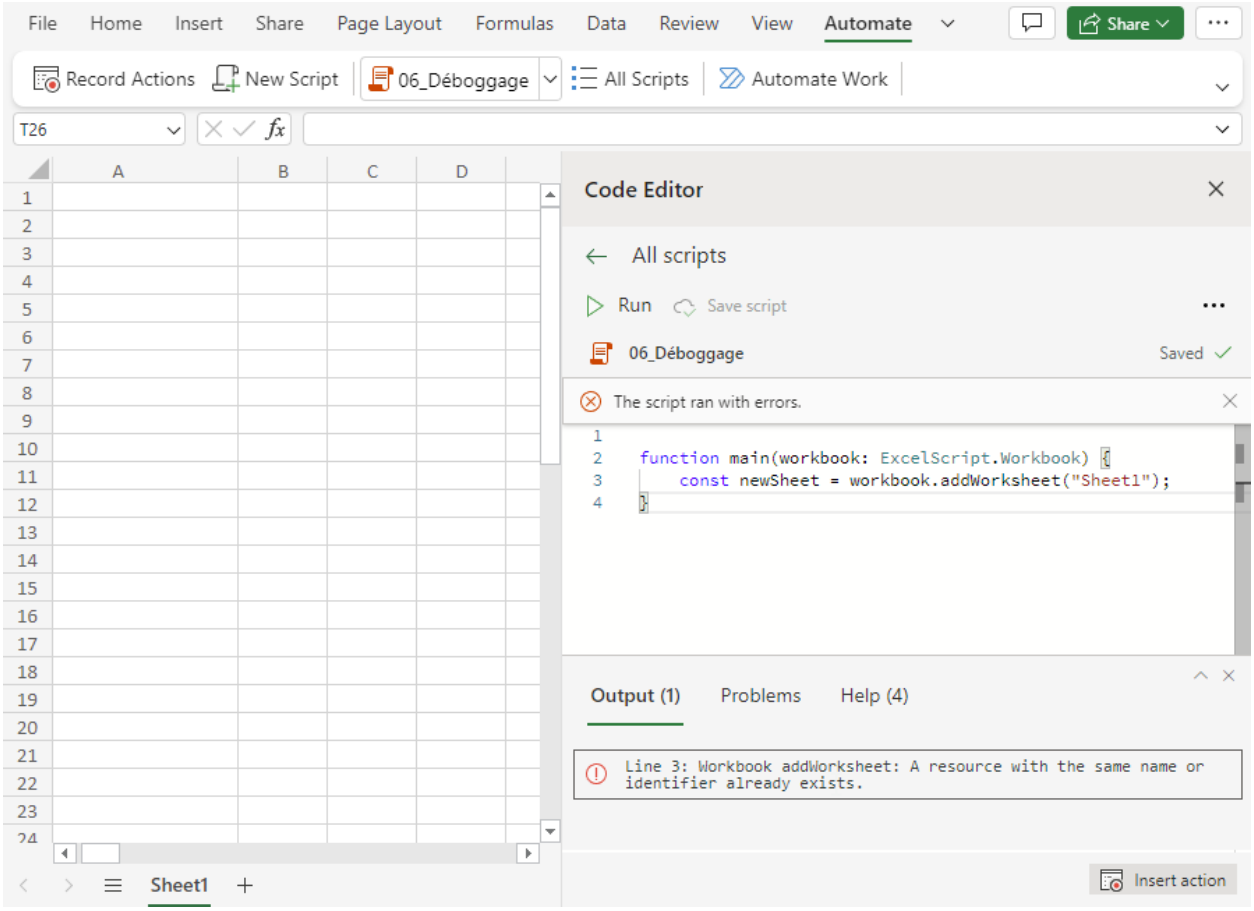
9.5 Débogage

Nous allons reproduire ici le cas fameux que nous voyons dans le cours V.B.A. et qui est la création de feuilles ayant un nom déjà existant!

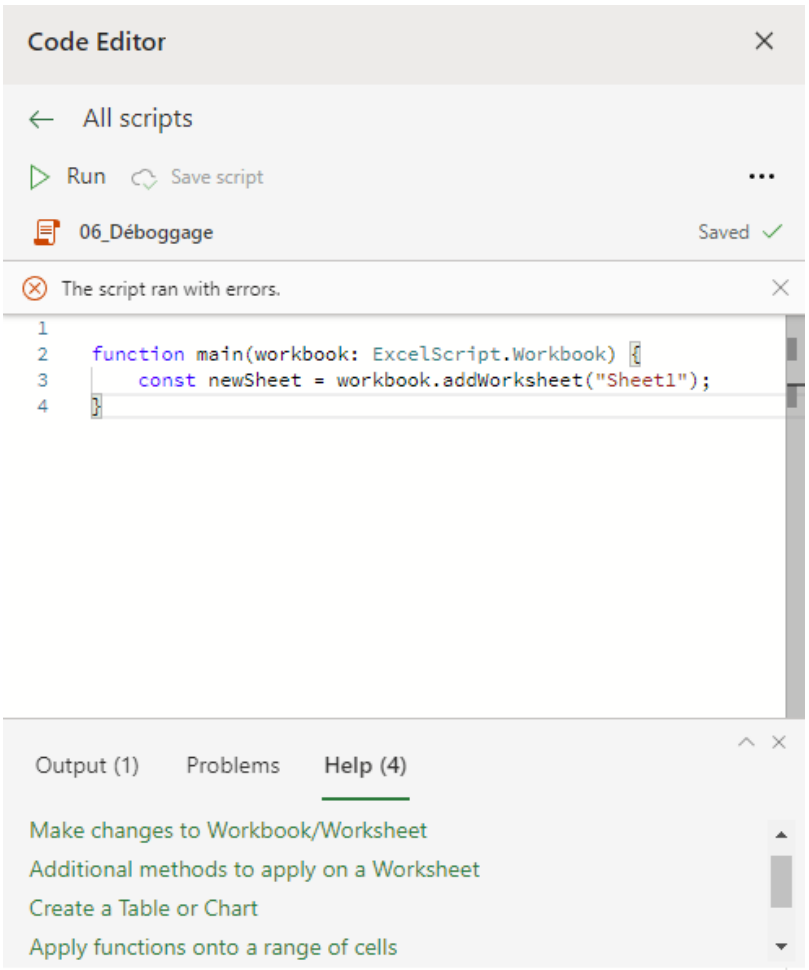
Considérons la situation suivante:



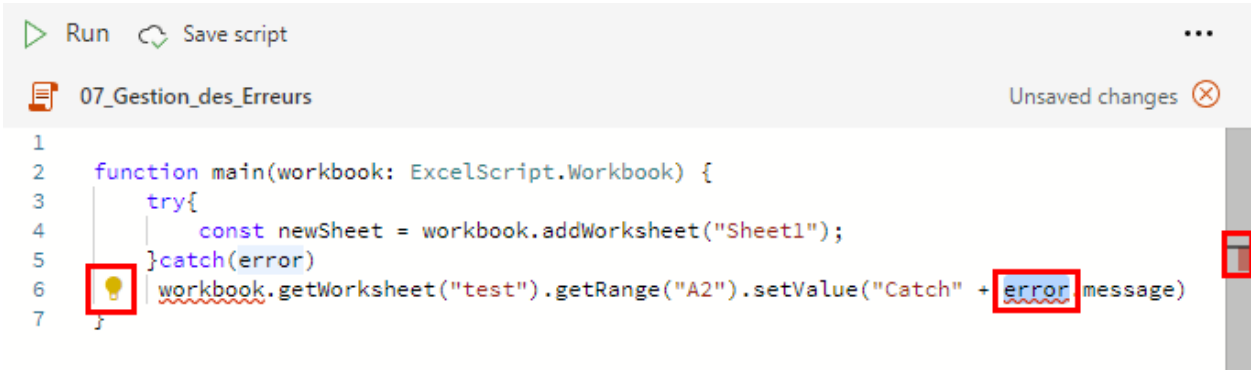
Si nous exécutons le code, nous avons:



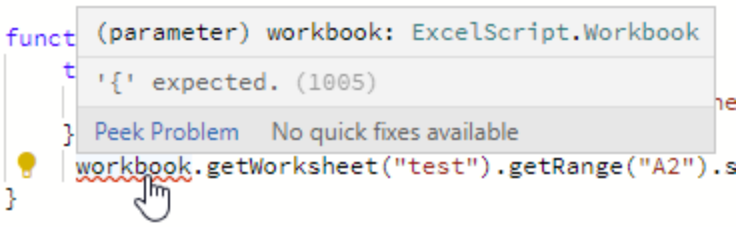
Nous voyons donc déjà que même dans ce cas simple, le message d'erreur est plus explicite. Si nous cliquons sur:



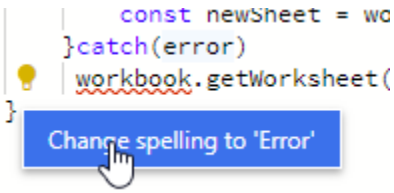
Voyons un cas où il va indiquer des erreurs mais toujours au mauvais endroit et le débugger va en réalité empirer la situation:



Comme nous le voyons il y a différentes aides pour tenter d'indiquer l'erreur:



Mais certaines indications sont aberrantes:



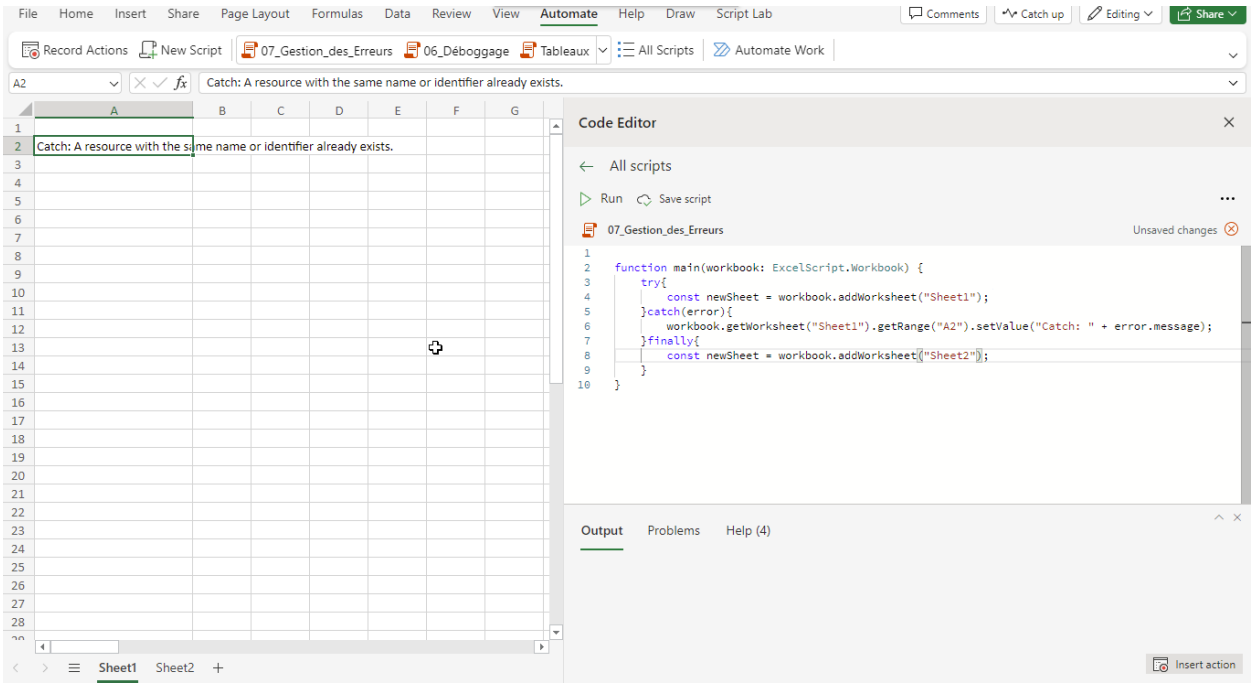
Maintenant voyons comment améliorer cela avec la gestion des erreurs qui fait l'objet de la prochaine section.

9.6 Gestion des erreurs

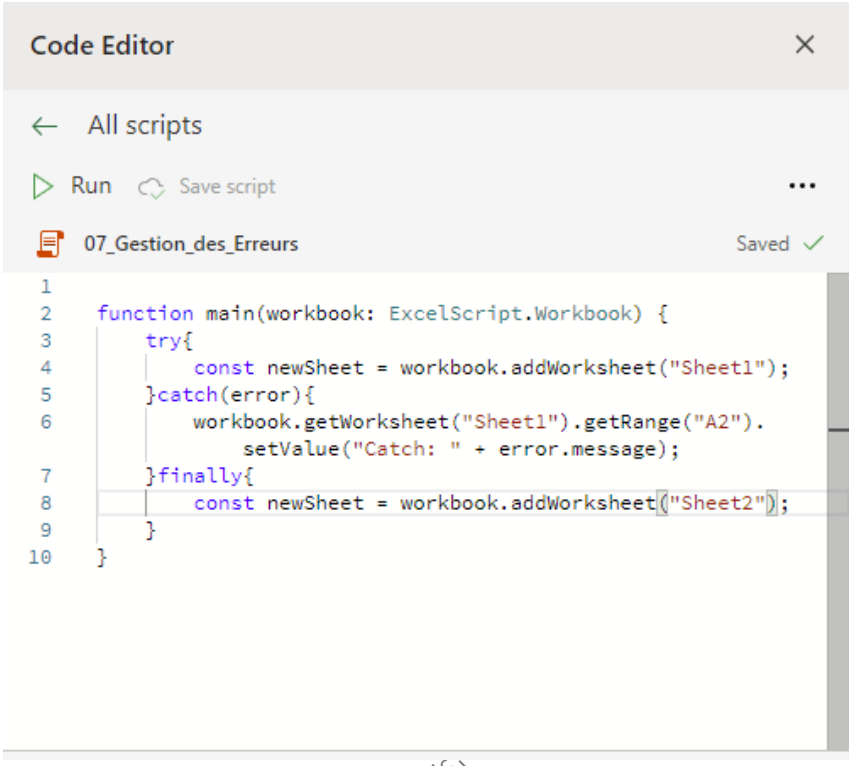
La gestion des erreurs est tellement pauvre à ce jour (2023) avec Office Script que cela en est presque exaspérant.

Public
Internal

Voyons un cas typique de provocation d'une erreur très connue chez les débutants du VBA:



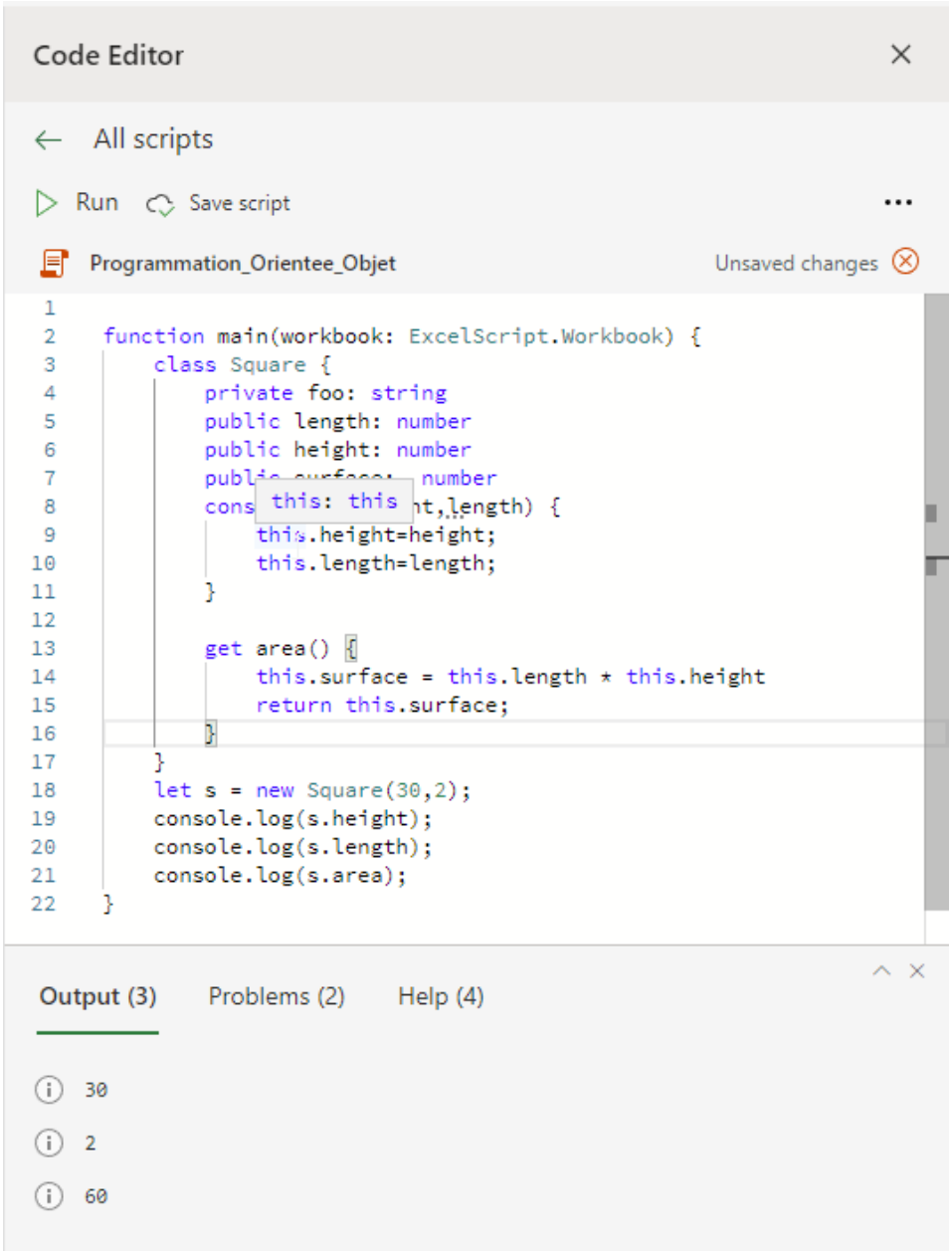
Soit avec un zoom:



Public
Internal

9.7 Programmation Orientée Objet

Cette section sera complétée quand j'aurai une demande pour une formation avancée Office Script. À ce jour, le seul objet que je suis arrivé à créer (et qui est incomplet) dans ce langage est le suivant:



Et des mystères de syntaxes et règles grammaticales subsistent encore pour moi… cette section de ce support de cours est donc encore à améliorer.

10 Userforms

Comme déjà mentionné plusieurs fois au début de ce livre, nous ne pouvons pas créer des userforms avec les Office Scripts encore en 2023… Il faut bricoler et dessiner des pseudo-formulaires sur les feuilles Microsoft Excel comme nous le faisions avec Excel 4.0 avant 1995…

Donc évidemment nous n'allons pour le coup pas pouvoir traiter de ce sujet puisqu'il n'existe pas!

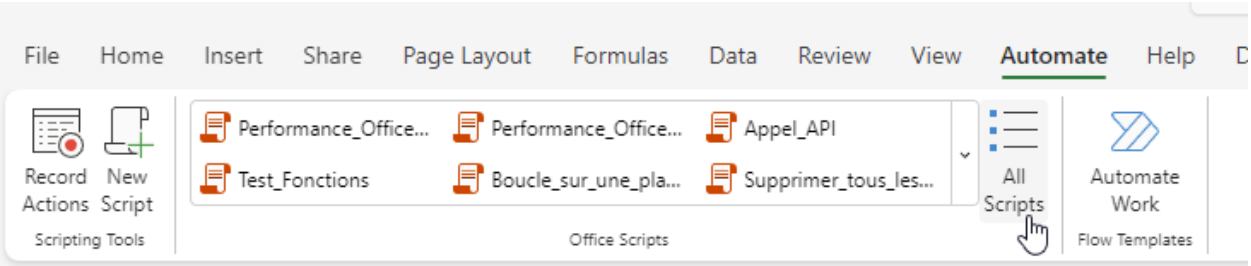
Public
Internal

11 Procédures et Fonctions Classiques

Comme pour le support de cours VBA, nous avons mis ici que des codes qui ne peuvent pas au jour où nous écrivons

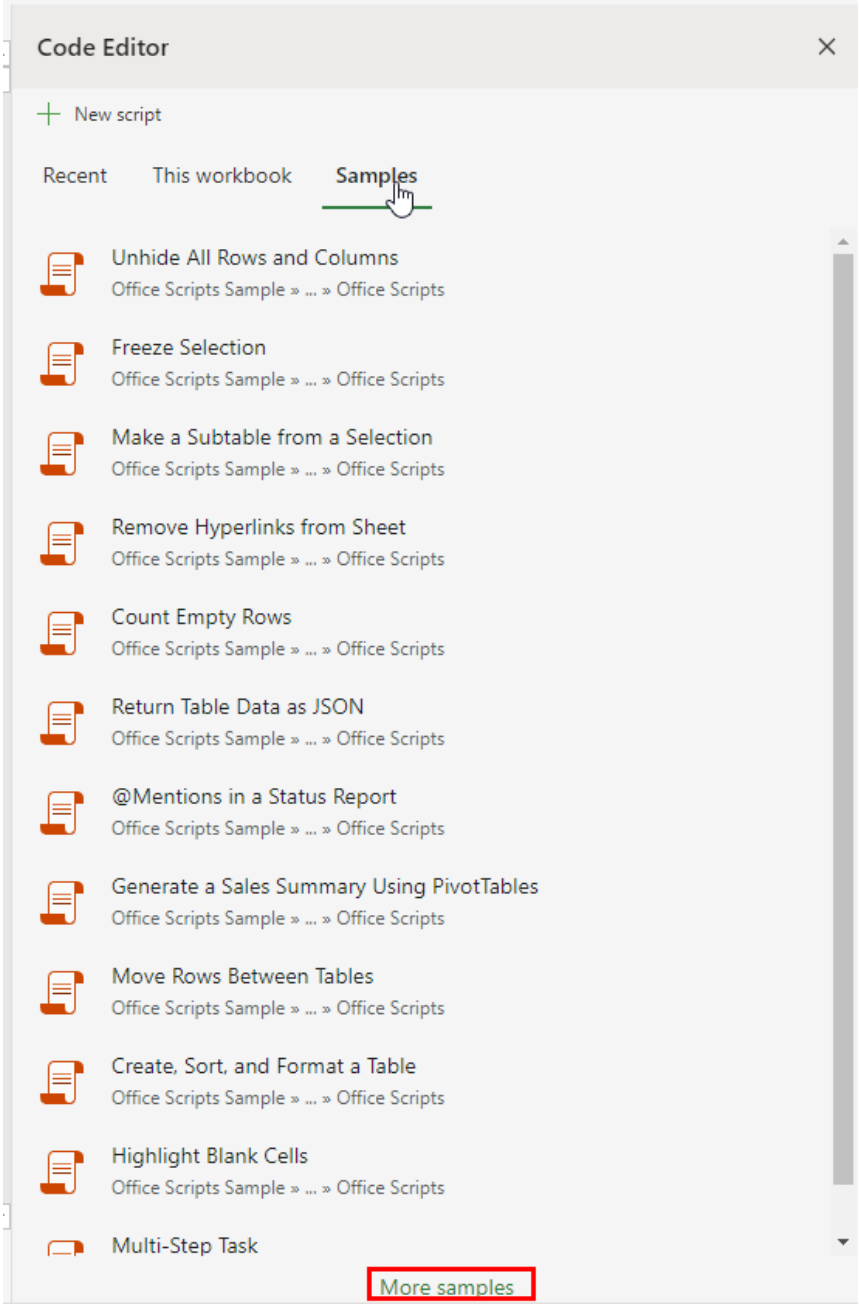
11.1 Modèles Office Script Microsoft

D'abord, ce qu'il faut savoir, c'est que quand vous cliquez sur **All Scripts**:

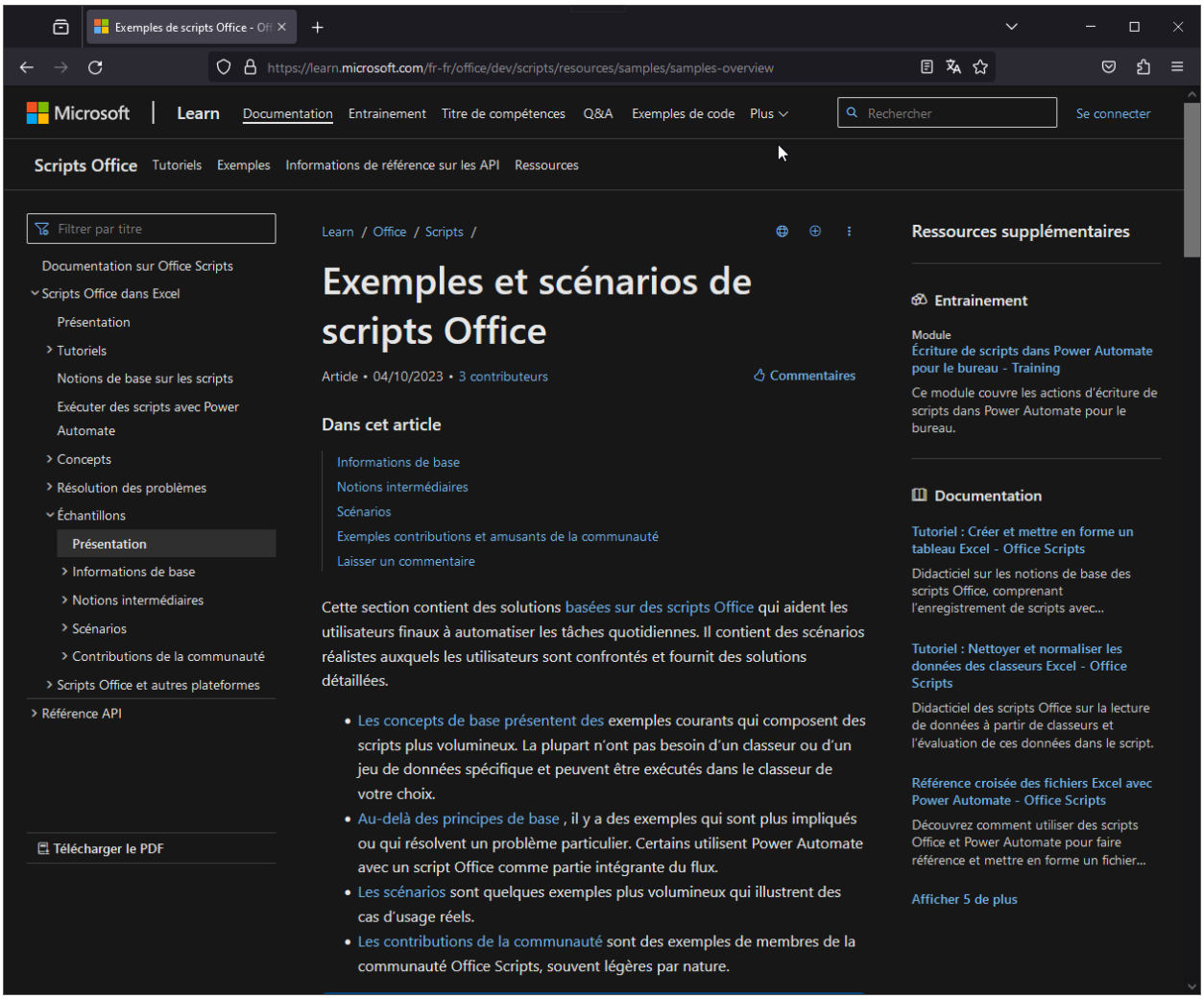


Nous avons alors:

Public Internal



Si nous cliquons sur **More Sample**s nous arrivons sur la page web suivante de Microsoft:

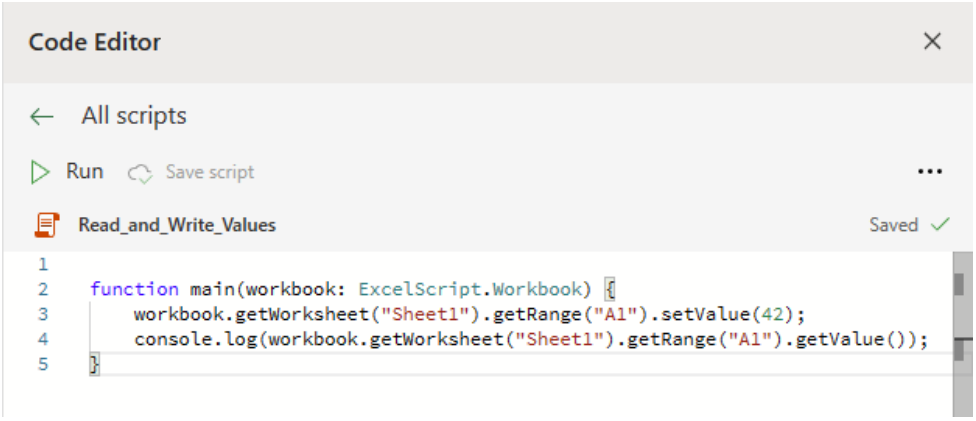

Exemples de scripts Office - Off
https://learn.microsoft.com/fr-fr/office/dev/scripts/resources/samples/samples-overview
Microsoft | Learn
Documentation
Entrainement
Titre de compétences
Q&A
Exemples de code
Plus
Rechercher
Se connecter
Scripts Office
Tutoriels
Exemples
Informations de référence sur les API
Ressources
Filtrer par titre
Documentation sur Office Scripts
Scripts Office dans Excel
Présentation
Tutoriels
Notions de base sur les scripts
Exécuter des scripts avec Power Automate
Concepts
Résolution des problèmes
Échantillons
Présentation
Informations de base
Notions intermédiaires
Scénarios
Contributions de la communauté
Scripts Office et autres plateformes
Référence API
Télécharger le PDF
Learn / Office / Scripts /
Exemples et scénarios de scripts Office
Article • 04/10/2023 • 3 contributeurs
Commentaires
Dans cet article
Informations de base
Notions intermédiaires
Scénarios
Exemples contributions et amusants de la communauté
Laisser un commentaire
Cette section contient des solutions basées sur des scripts Office qui aident les utilisateurs finaux à automatiser les tâches quotidiennes. Il contient des scénarios réalistes auxquels les utilisateurs sont confrontés et fournit des solutions détaillées.
Les concepts de base présentent des exemples courants qui composent des scripts plus volumineux. La plupart n'ont pas besoin d'un classeur ou d'un jeu de données spécifique et peuvent être exécutés dans le classeur de votre choix.
Au-delà des principes de base , il y a des exemples qui sont plus impliqués ou qui résolvent un problème particulier. Certains utilisent Power Automate avec un script Office comme partie intégrante du flux.
Les scénarios sont quelques exemples plus volumineux qui illustrent des cas d'usage réels.
Les contributions de la communauté sont des exemples de membres de la communauté Office Scripts, souvent légères par nature.
Ressources supplémentaires
Entrainement
Module
Écriture de scripts dans Power Automate pour le bureau - Training
Ce module couvre les actions d'écriture de scripts dans Power Automate pour le bureau.
Documentation
Tutoriel : Créer et mettre en forme un tableau Excel - Office Scripts
Didacticiel sur les notions de base des scripts Office, comprenant l'enregistrement de scripts avec...
Tutoriel : Nettoyer et normaliser les données des classeurs Excel - Office Scripts
Didacticiel des scripts Office sur la lecture de données à partir de classeurs et l'évaluation de ces données dans le script.
Référence croisée des fichiers Excel avec Power Automate - Office Scripts
Découvrez comment utiliser des scripts Office et Power Automate pour faire référence et mettre en forme un fichier...
Afficher 5 de plus

11.2 Lire/Écrire une valeur dans une cellule

Il existe comme nous le savons en VBA plusieurs manière de lire/écrire le contenu d'une cellule.

11.2.1 Par sa référence

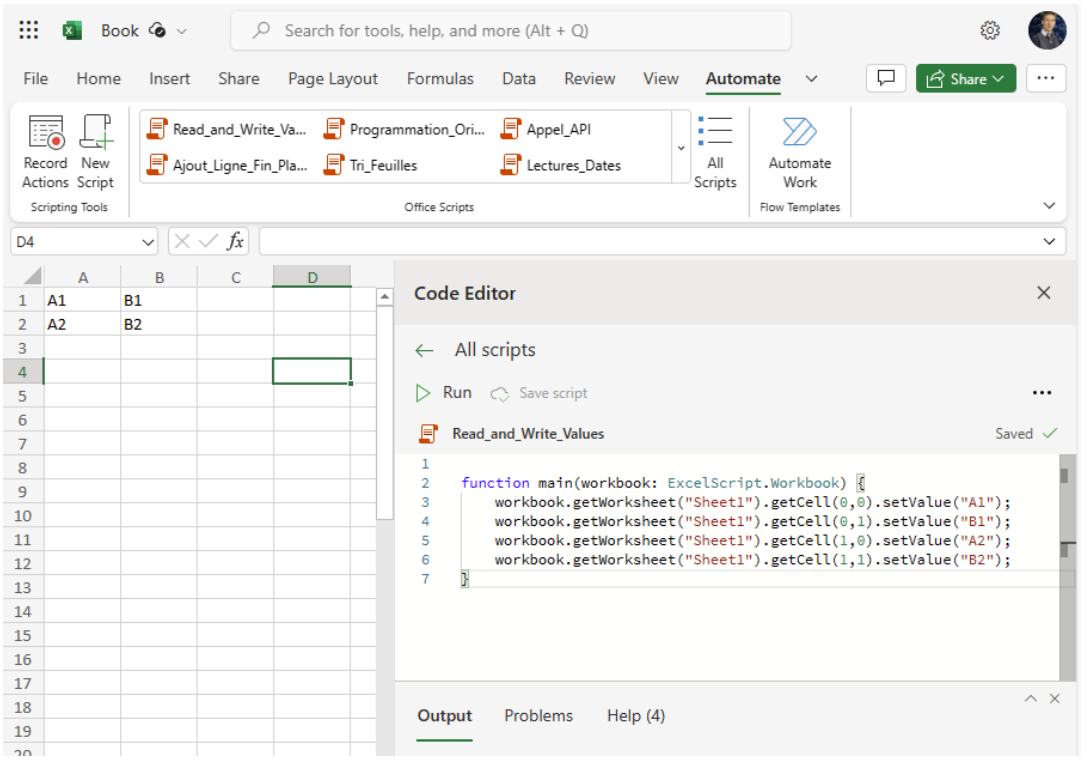
Le code Office Script pour lire ou écrire une valeur dans une cellule par son libellé est simplement:



11.2.2 Par son index

Public External

Le code Office Script pour écrire une valeur dans une cellule par son numéro d'index est simplement:



Et voilà!

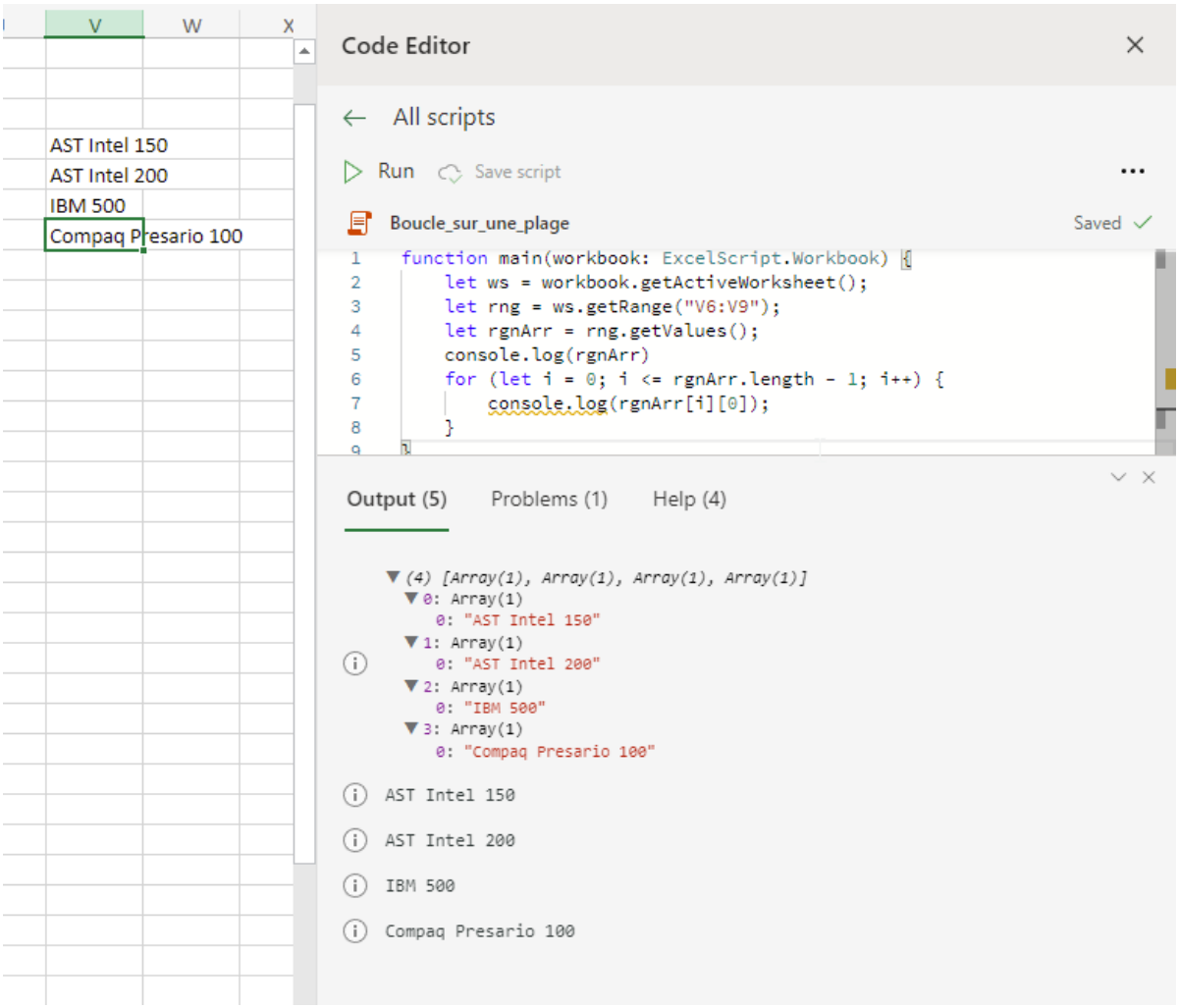
Public
Internal

11.3 Lire un nombre sous forme de texte dans un cellule et le coercer en nombre

To Do

11.4 Boucler dans un plage de cellules donnée

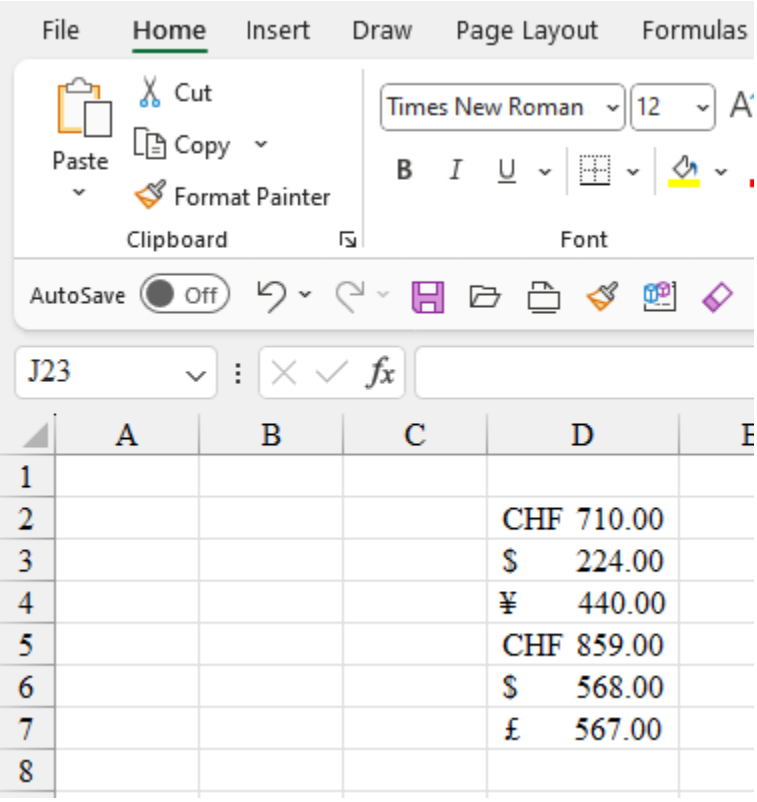
C'est un autre très grand classique et l'application la plus connue des boucles:



Et voilà!

11.5 Boucler dans un plage de cellules donnée pour identifier des formats

Considérons la petite plage suivante:

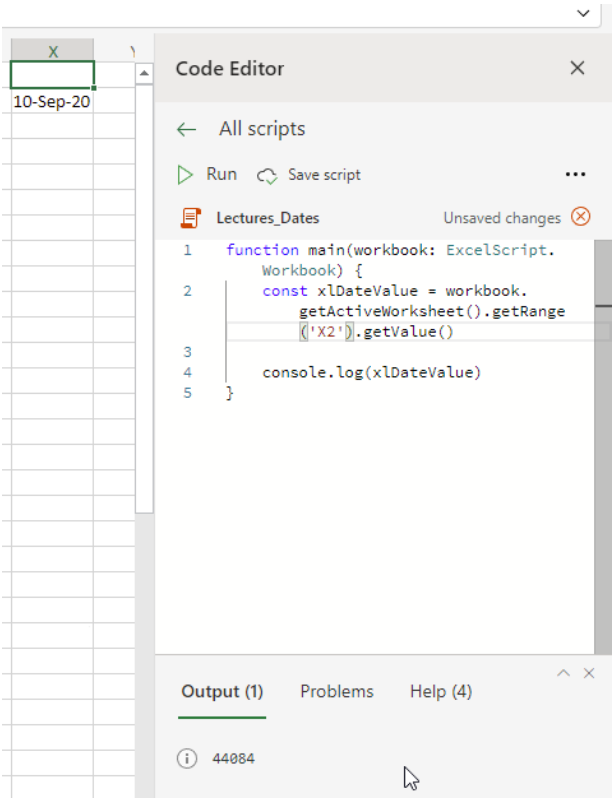


Nous voulons mettre avec un fond jaune les cellules ayant un format incluant un "$".

Pour cela, nous créons le script suivant:

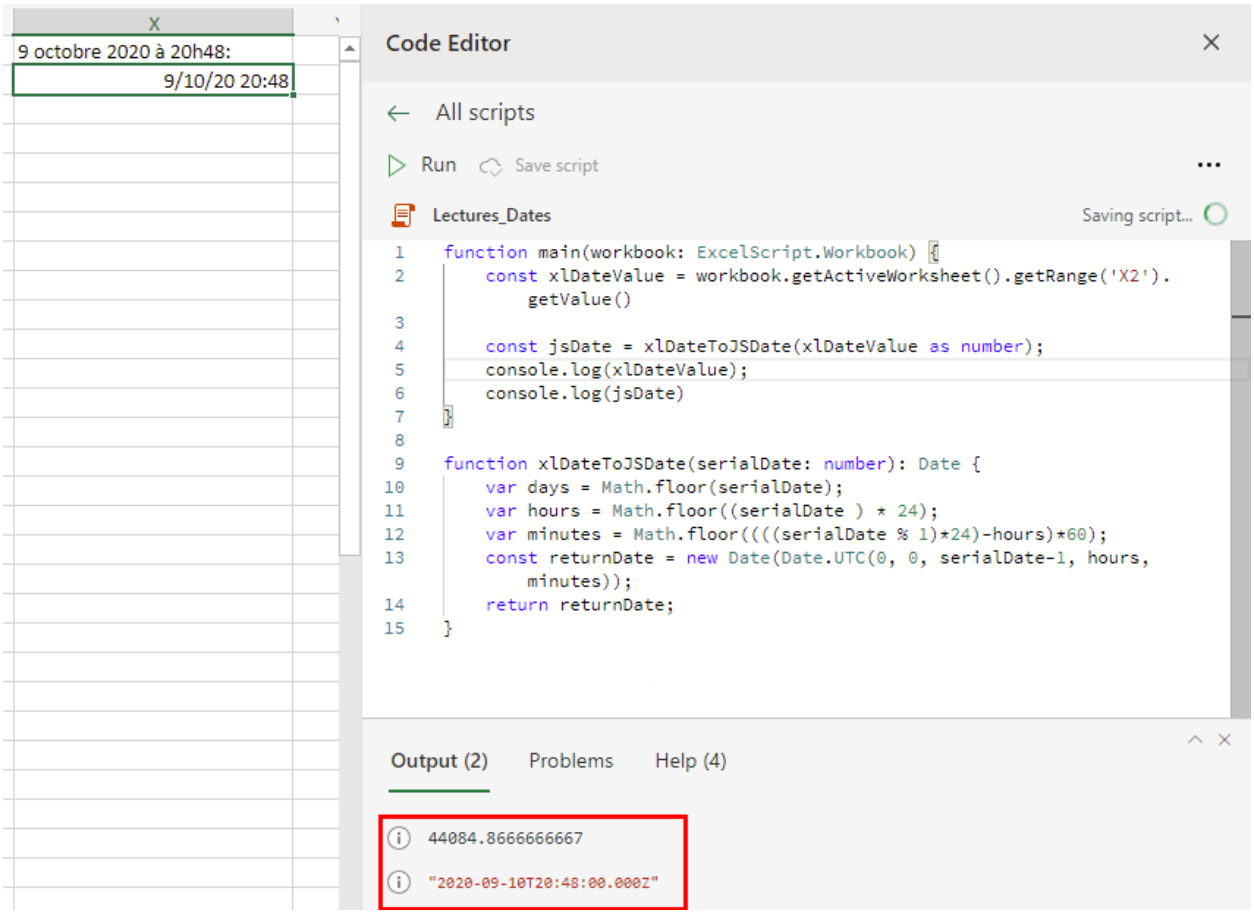
11.6 Afficher une date correctement dans la console

C'est une demande fréquente et connue. Considérons que nous avons dans la cellule X2 la date visible ci-dessous:



Nous voyons que la console nous donne le nombre de jours depuis le 1er Janvier 1901. C'est pas génial en termes de débogage. Comment remettre cela correctement?

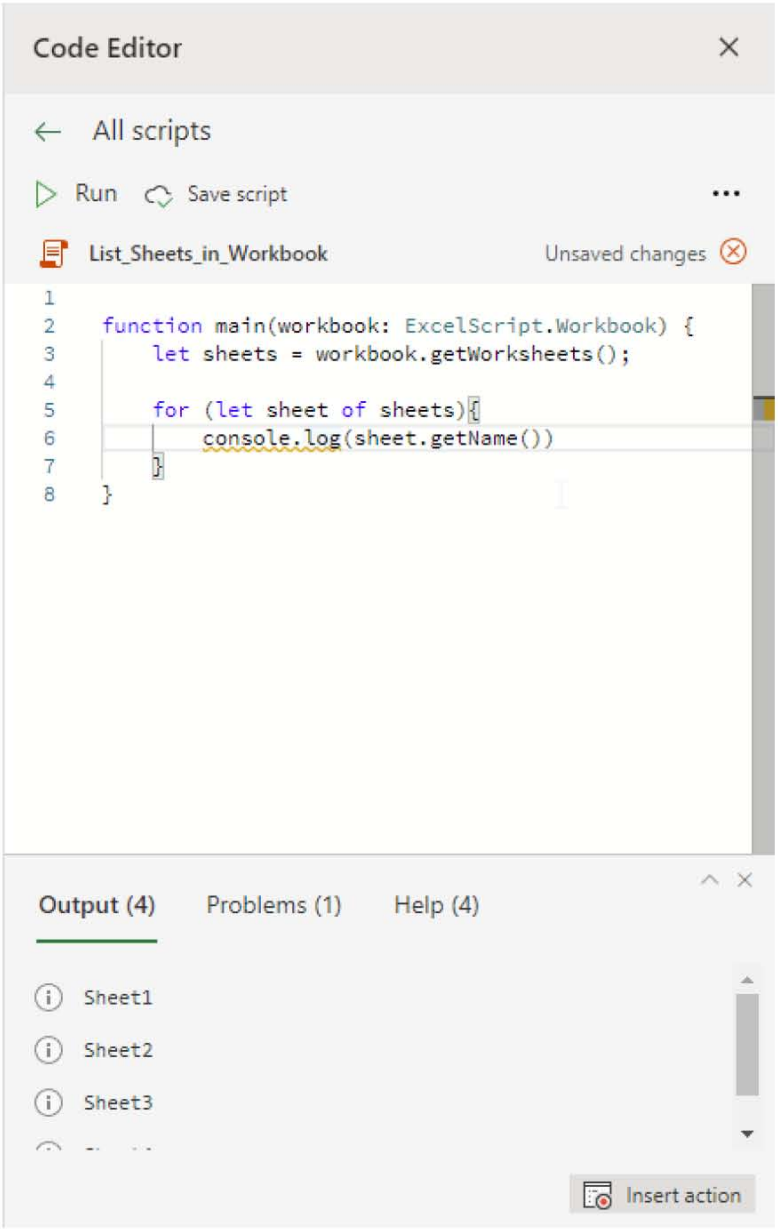
En création la fonction suivante (notez que nous avons changé la date et l'heure dans la cellule XS):



Ce qui est bien mieux!

11.7 Lister les feuilles d'un fichier

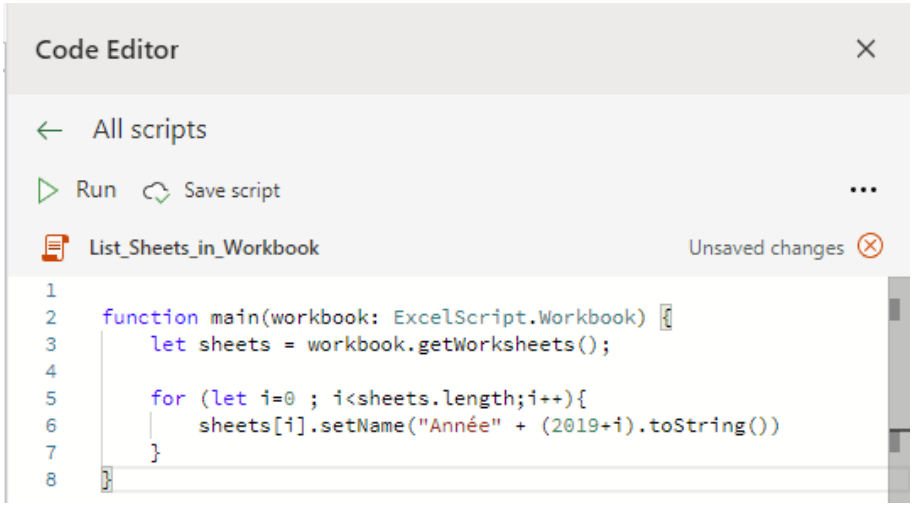
C'est évidemment un autre grand classique de parcourir la collection des feuilles:



Et voilà!

11.8 Renommer toutes les feuilles

Voici un petit renommage de feuilles avec en plus une concaténation assez typique :

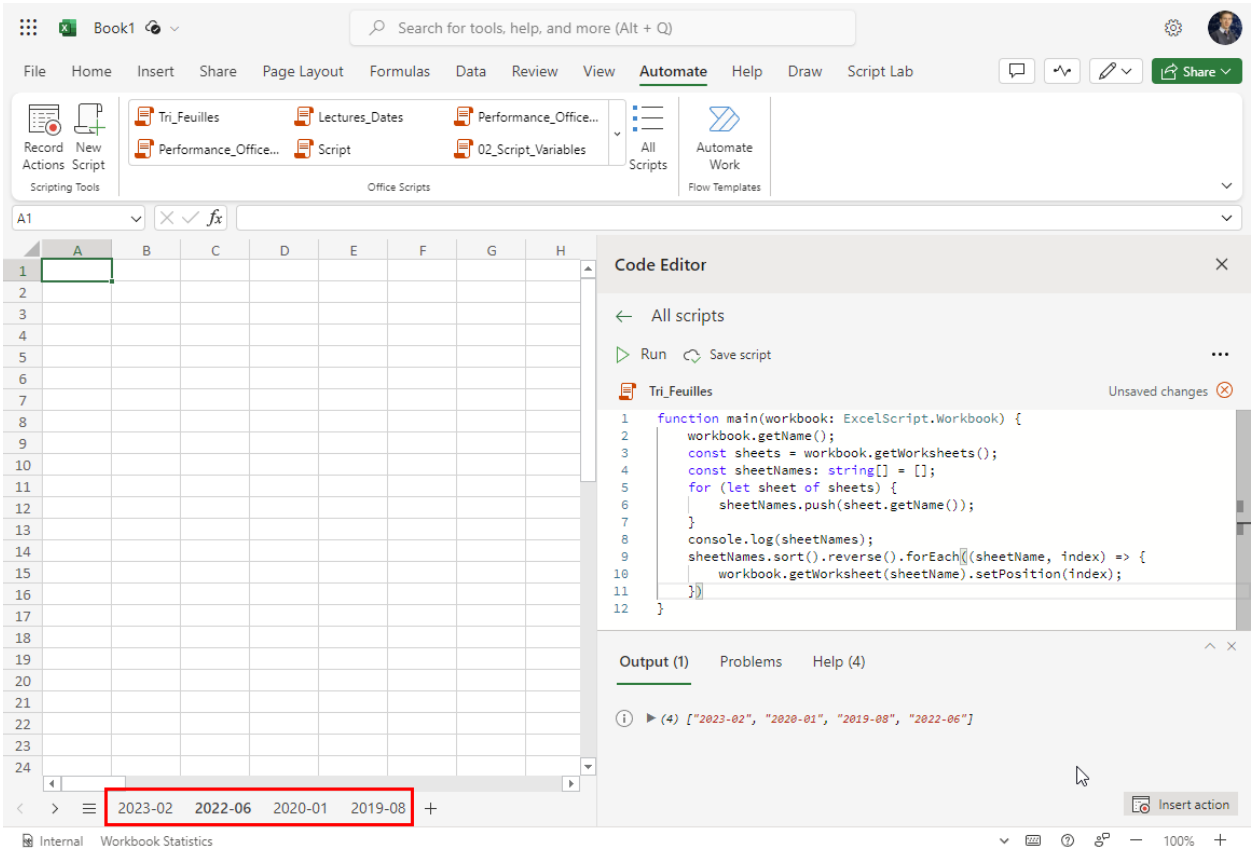


Et voilà! Rien de compliqué.

Public
Internal

11.9 Trier les feuilles

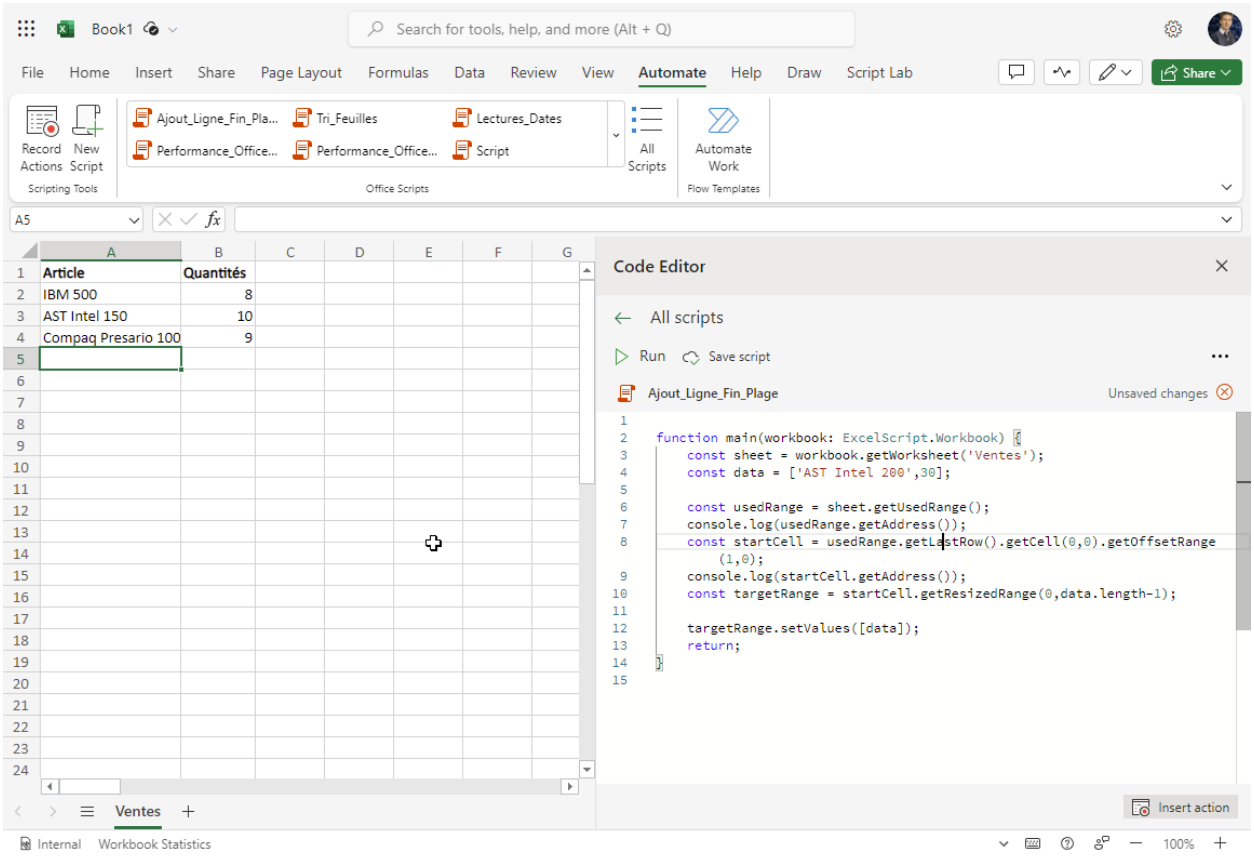
Prochain cas ultra standard et connu en VBA mais en son équivalent Office Script, le tri des feuilles:



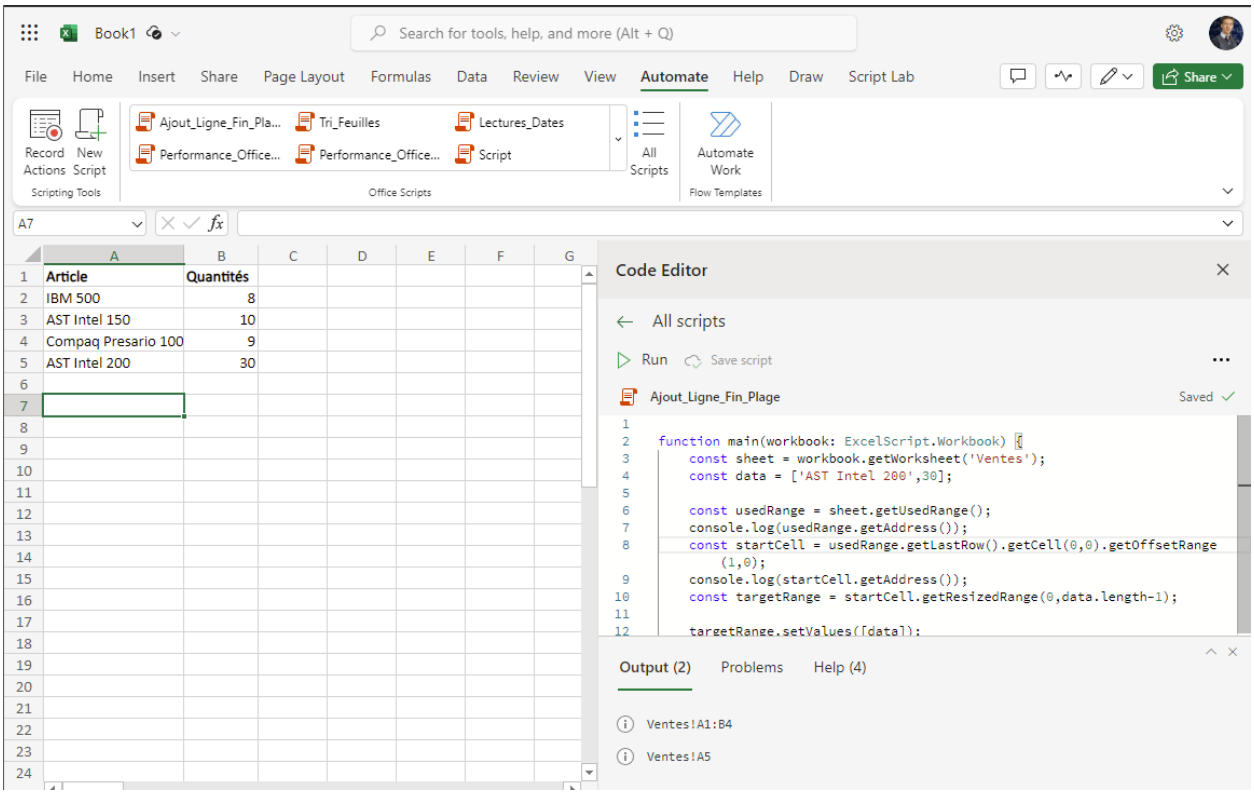
Et voilà!

11.10 Ajouter une ligne à la fin d'une table

Encore une autre cas super connu en VBA dont voici l'équivalent en Office Script qui est l'ajout d'une ligne de données à la fin d'une liste Excel:



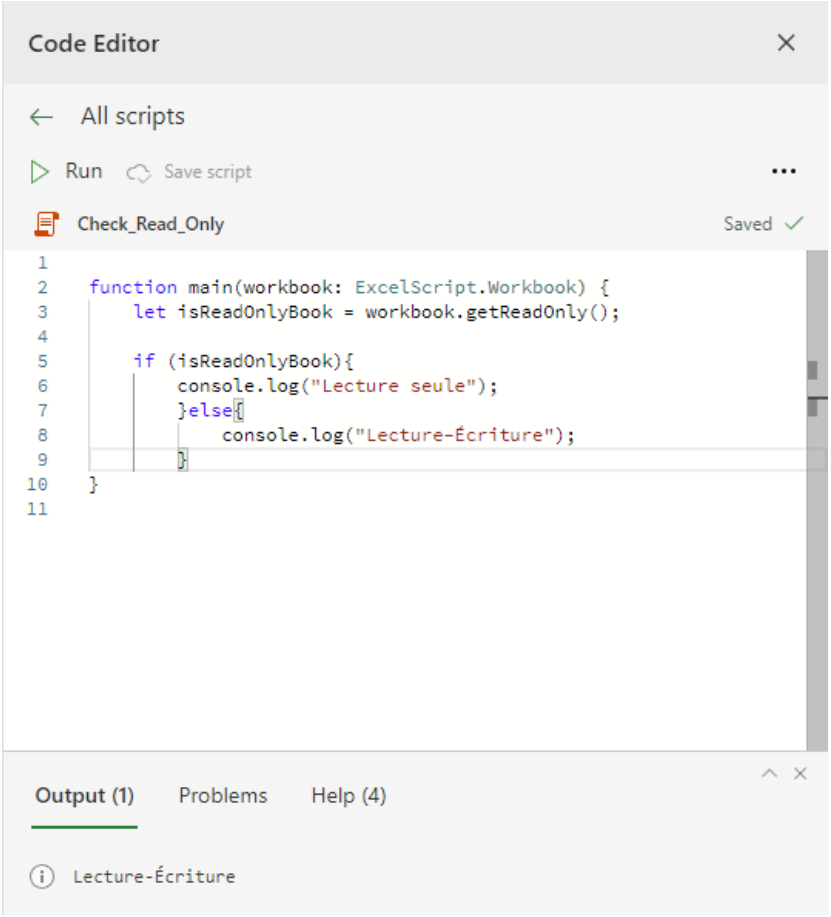
Ce qui donne une fois exécuté:



Public Internal

11.11 Vérifier si un fichier est en lecture seule

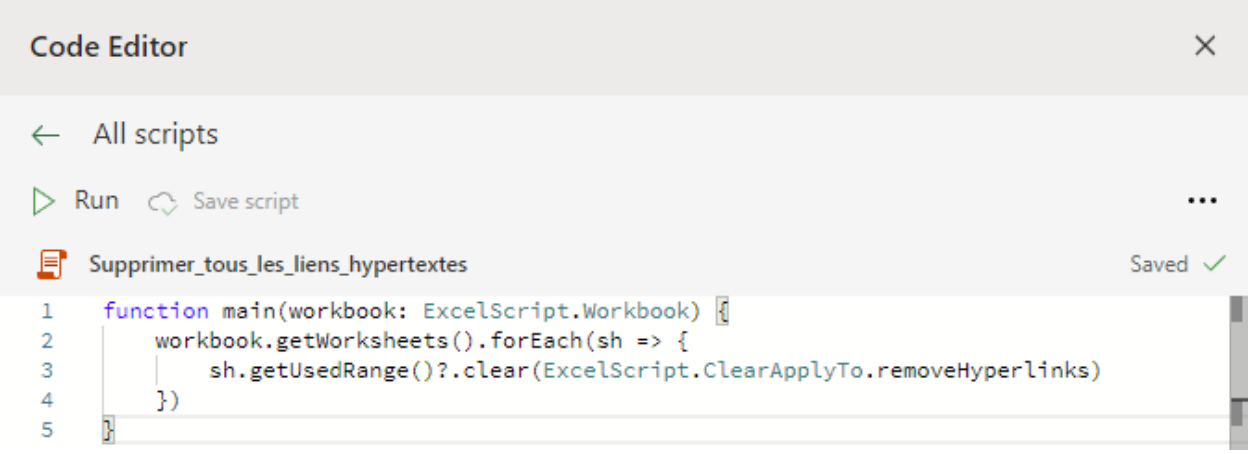
Un autre grand classique en VBA dont voici l'équivalent en Office Script:



Et voilà!

11.12 Supprimer tous les liens hypertextes d'un classeur

Encore un autre grand classique VBA mais dont voici l'équivalent en Office Script:



11.13 Créer une table des matières

Voici un script Office (pris de https://jkp-ads.com/articles/excel-office-script-examples.asp) qui en produit un. Il génère une liste de toutes les feuilles de calcul et de tous les graphiques, avec des hyperliens pointant vers leurs emplacements:

Public Internal

```
function main(workbook: ExcelScript.Workbook) {

    if (workbook.getWorksheet("ToC")) {
        let temp = workbook.getWorksheet("ToC")
        temp.delete();
    }

    let allSheets = workbook.getWorksheets();
    let tocSheet = workbook.addWorksheet("ToC");
    let reportCell = tocSheet.getRange("B2");
    let newTable = addTocTable("Worksheets", workbook, reportCell);
    reportCell = reportCell.getOffsetRange(2, 0);
    reportCell.getResizedRange(0, 1).setValues([
        ["Worksheet", "Link"]
    ]);

    for (let i = 0; i < allSheets.length; i++) {
        reportCell.getOffsetRange(i + 1, 0).setValue(allSheets[i].getName());
    }
    reportCell.getOffsetRange(1, 1).setFormulaR1C1("=HYPERLINK(\"#'\"&RC[-1]&\"'!A1\",RC[-1])");

    reportCell = reportCell.getOffsetRange(newTable.getRowCount() + 2, 0)
    newTable = addTocTable("Charts", workbook, reportCell);
    reportCell = reportCell.getOffsetRange(2, 0)
    reportCell.getResizedRange(0, 3).setValues([
        ["Worksheet", "Name", "Location", "Link"]
    ]);
    let k = 0;
    for (let i = 0; i < allSheets.length; i++) {
        let allCharts = allSheets[i].getCharts();
```

```
        for (let j = 0; j < allCharts.length; j++) {
            k++;
            let theSheet = allSheets[i];
            reportCell.getOffsetRange(k, 0).setValue(allSheets[i].getName());
            reportCell.getOffsetRange(k, 1).setValue(allCharts[j].getName());
            let topLeftCell = getCellUnderChart(allCharts[j], allSheets[i]);
            let topLeftAddress = topLeftCell.getAddress();
            if (topLeftAddress.charAt(0) == "'") {
                topLeftAddress = "'" + topLeftAddress;
            }
            reportCell.getOffsetRange(k, 2).setFormula(topLeftAddress);
        }
    }
    reportCell.getOffsetRange(1, 3).setFormulaR1C1("=HYPERLINK(\"#\"&RC[-1],\"Link\")");

    tocSheet.activate();
}

function addTocTable(caption: String, workbook: ExcelScript.Workbook, location:
ExcelScript.Range): ExcelScript.Table {
    location.setValue(caption);
    location.getResizedRange(0, 1).setPredefinedCellStyle("Heading 1");
    location = location.getOffsetRange(2, 0);
    let newTable = workbook.addTable(location, true);
    return newTable;
}

function getCellUnderChart(cht: ExcelScript.Chart, ws: ExcelScript.Worksheet): ExcelScript.Range
{
    let topLeftCell = ws.getRange("A1");
    do {
        topLeftCell = topLeftCell.getOffsetRange(0, 1)
    }
    while (topLeftCell.getLeft() < cht.getLeft());
    do {
        topLeftCell = topLeftCell.getOffsetRange(1, 0)
    }
    while (topLeftCell.getTop() < cht.getTop());

    return topLeftCell.getOffsetRange(-1, -1);
}
```

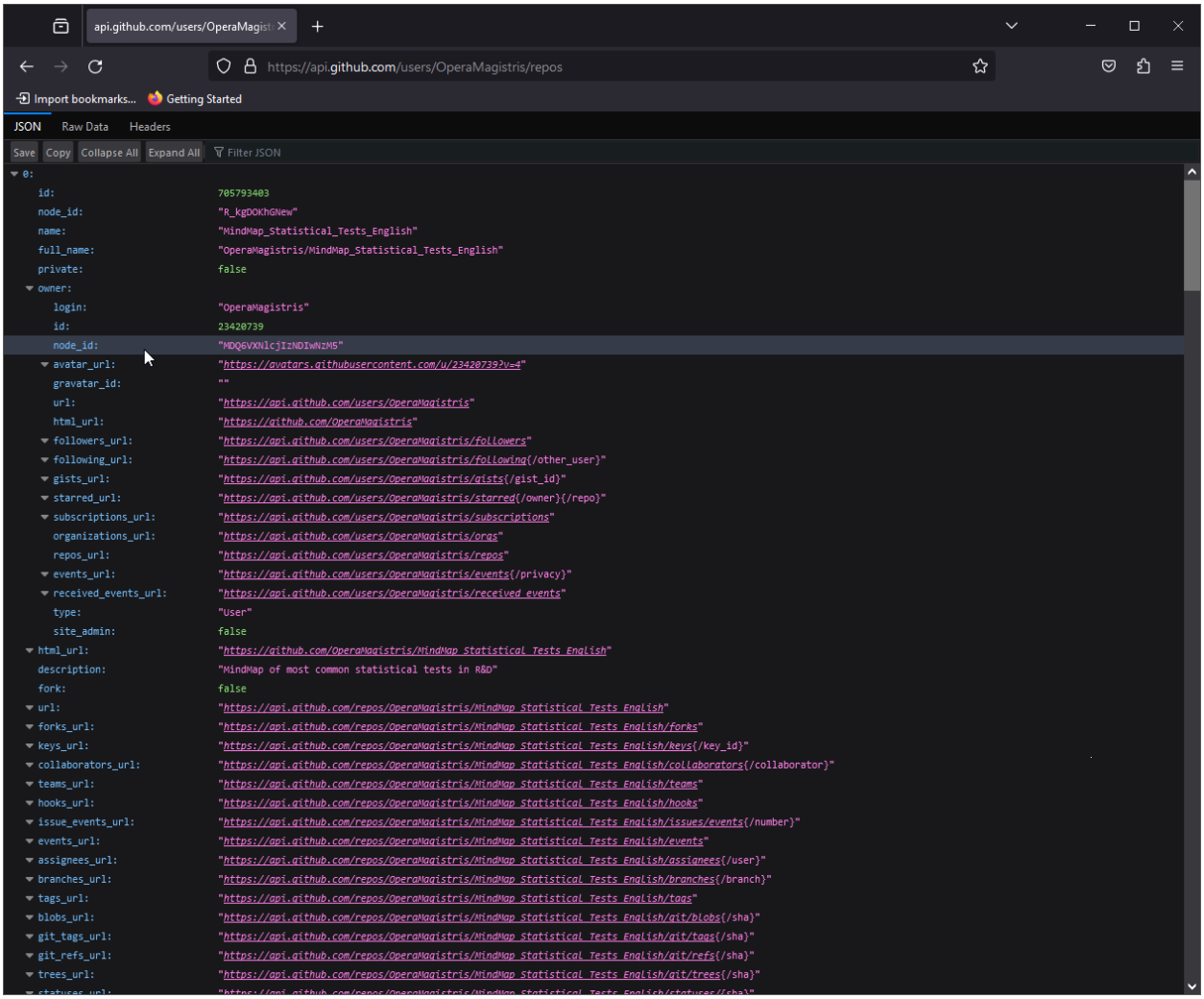
Public Internal

11.14 Appeler une API

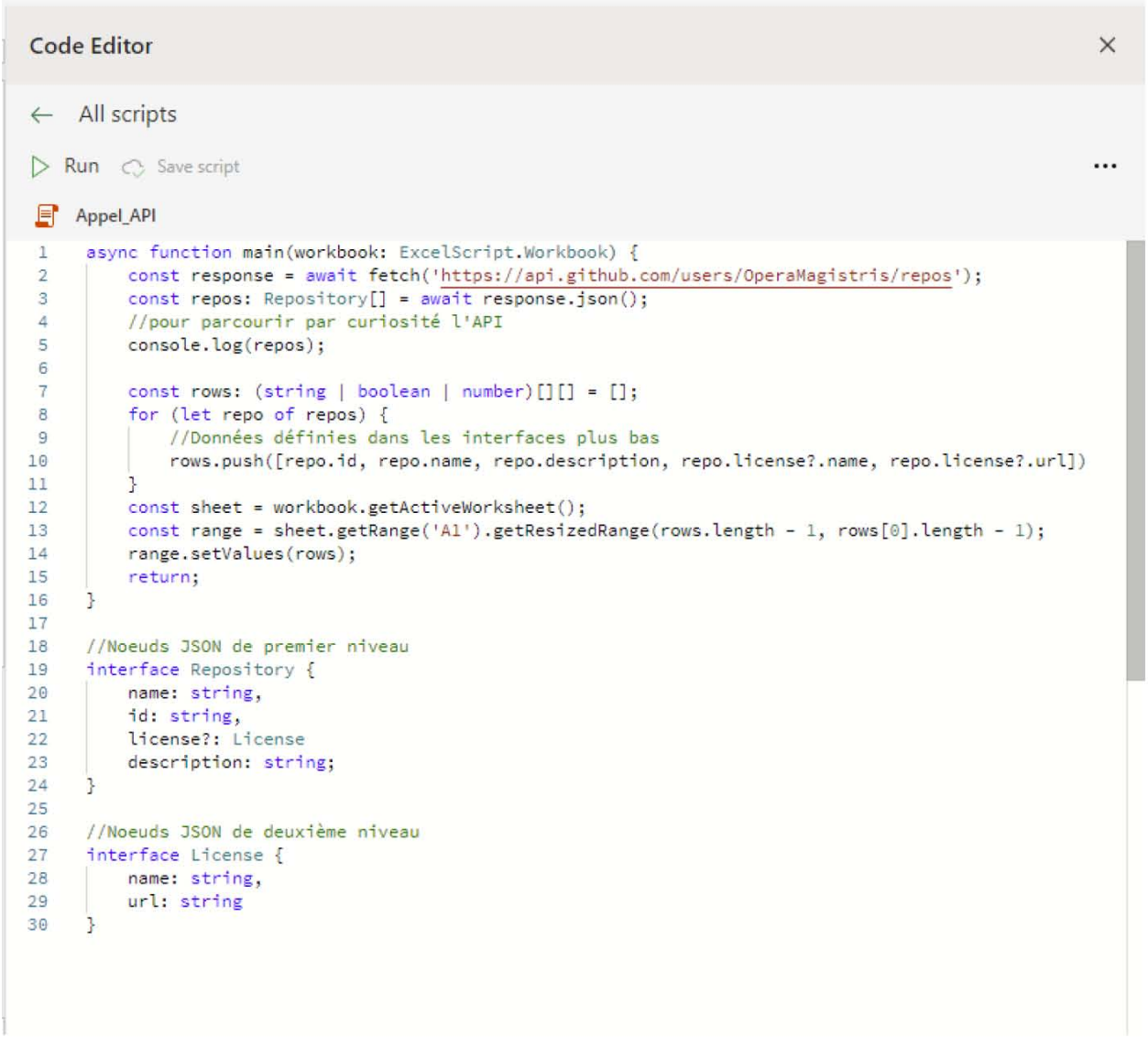
Considérons l'API suivante:

https://api.github.com/users/OperaMagistris/repos

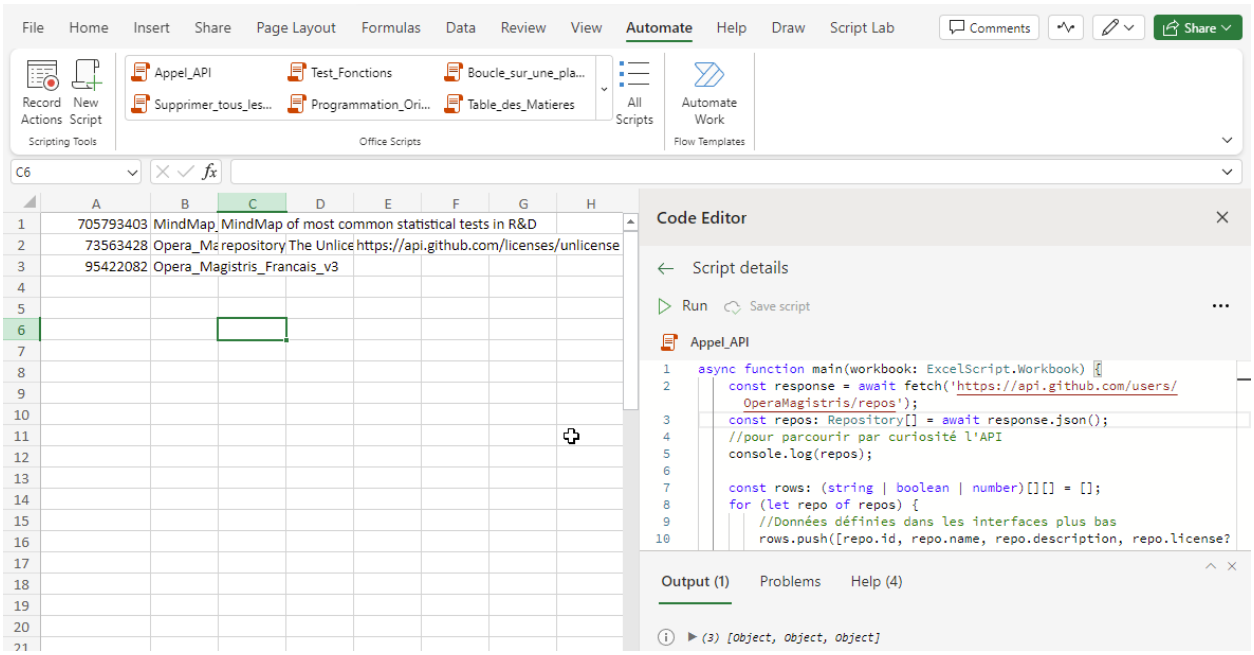
Qui donne:



Nous allons appeler certains nœuds de cet API avec l'Office Script suivant:



Ce qui donnera:



Et voilà!

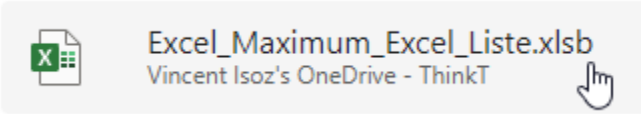
Public
Internal

12 Performance

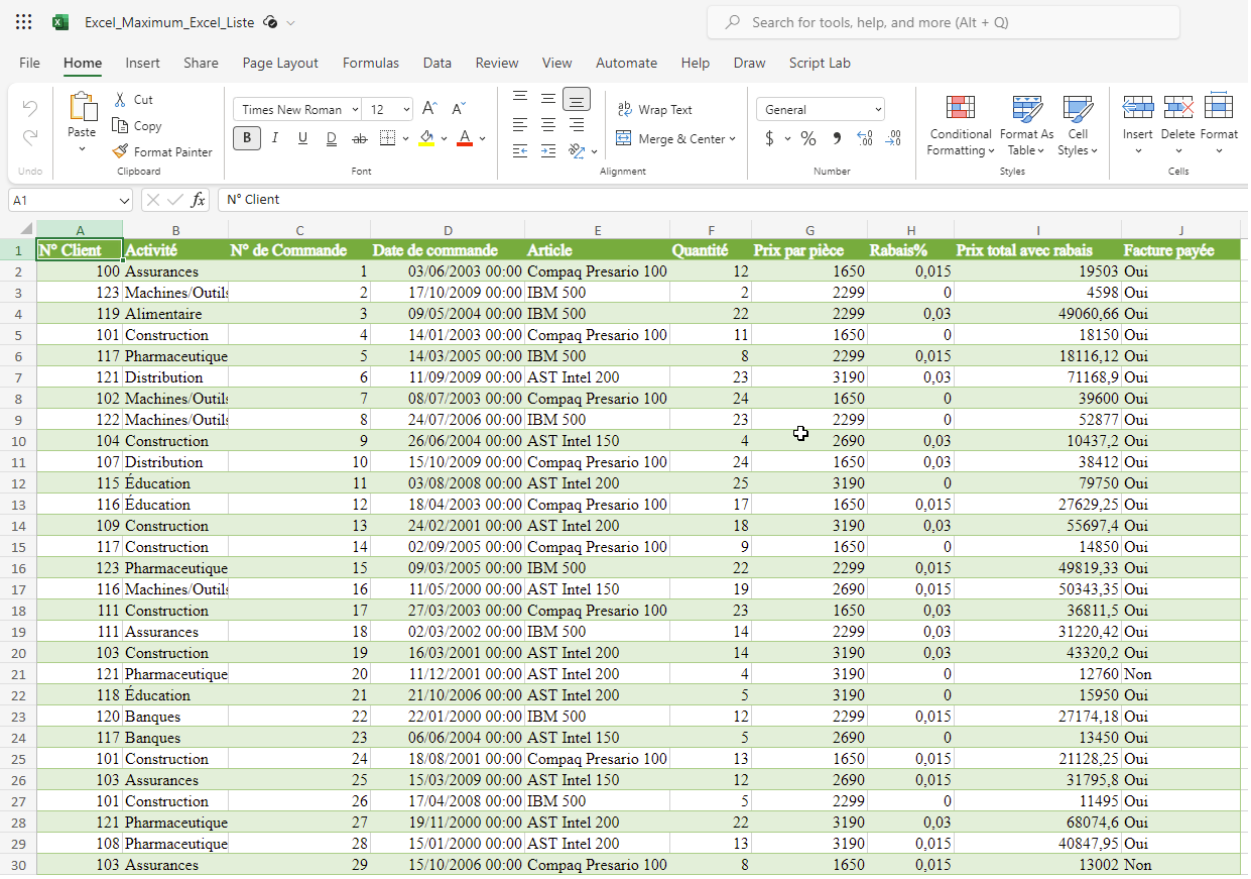
Vous pouvez appliquer des techniques simples pour réduire la communication vers le cloud. Les modèles suivants vous aident à accélérer vos scripts.

- Lisez les données du classeur une seule fois au lieu de plusieurs fois dans une boucle.
- Supprimez les instructions inutiles console.log .
- Évitez d'utiliser des blocs try/catch.

Voyons un exemple. Ouvrez le fichier:

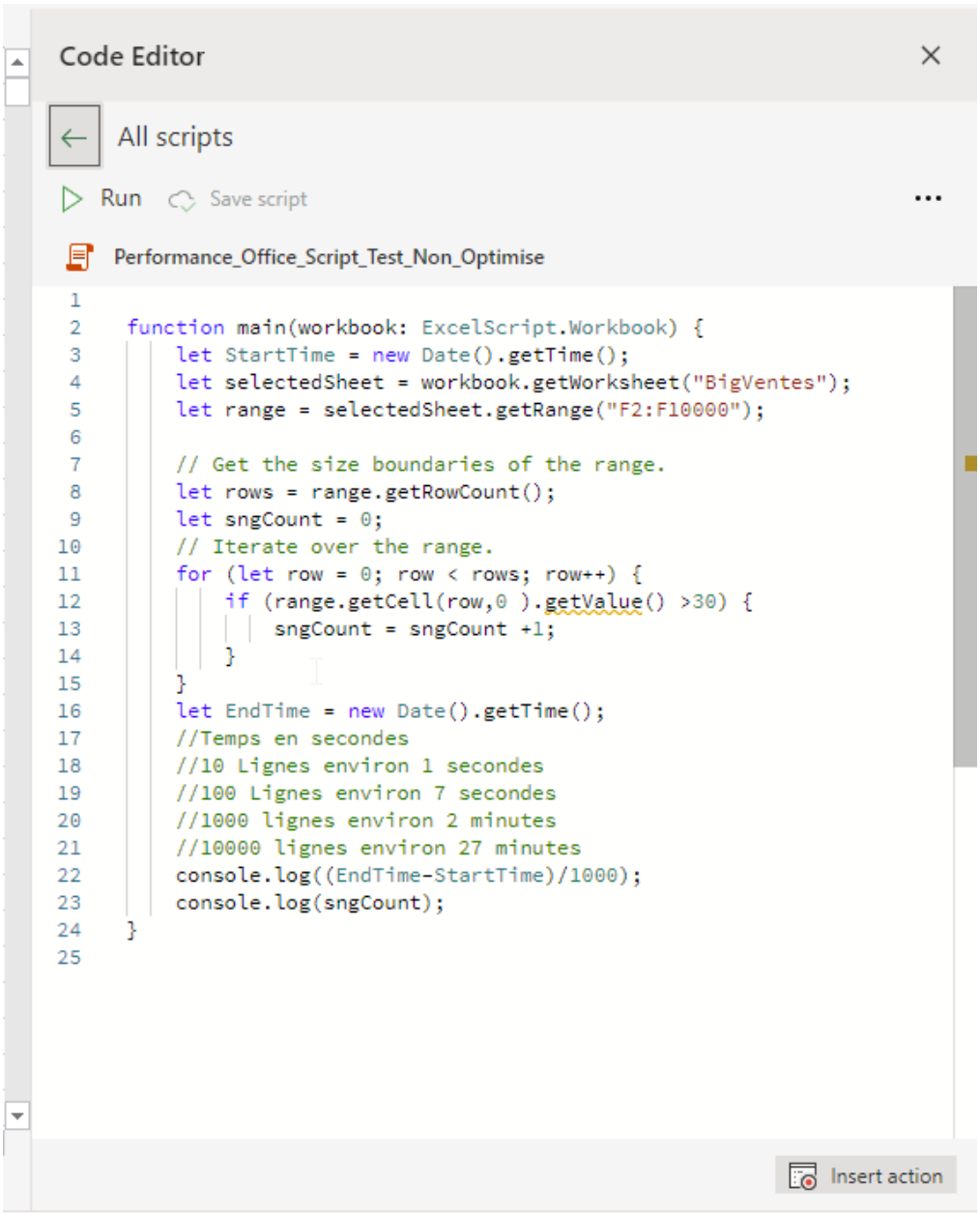


Ce qui donnera:

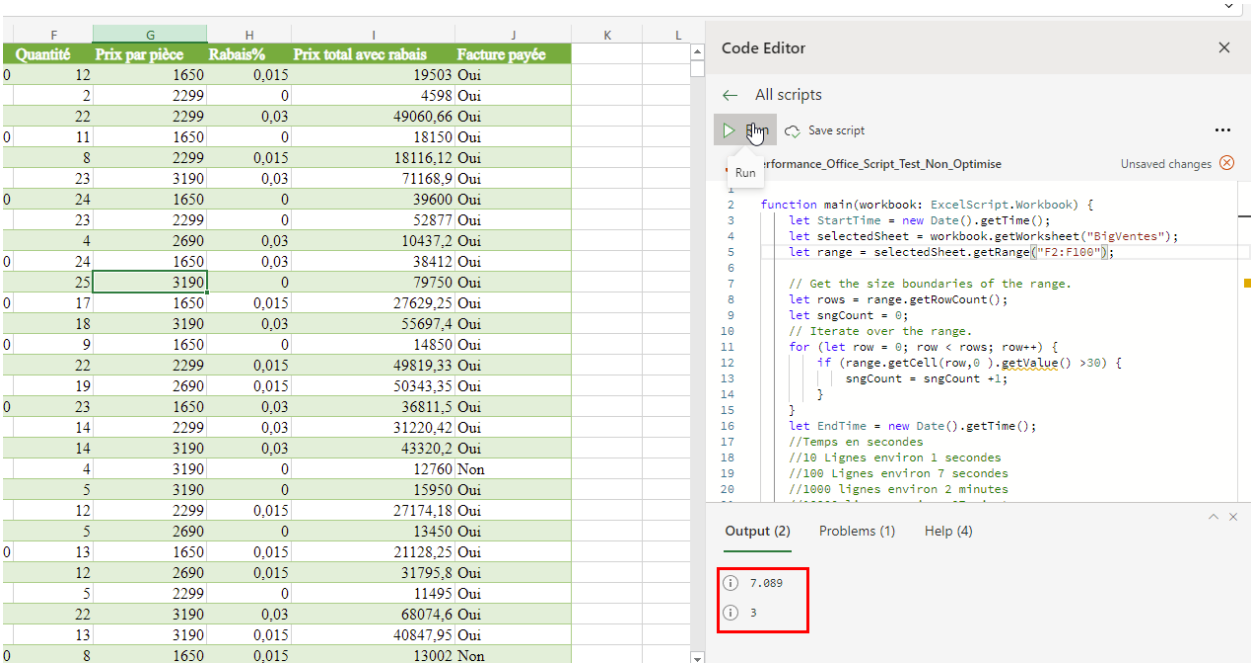


N° Client	Activité	N° de Commande	Date de commande	Article	Quantité	Prix par pièce	Rabais%	Prix total avec rabais	Facture payée
100	Assurances	1	03/06/2003 00:00	Compaq Presario 100	12	1650	0,015	19503	Oui
123	Machines/Outil	2	17/10/2009 00:00	IBM 500	2	2299	0	4598	Oui
119	Alimentaire	3	09/05/2004 00:00	IBM 500	22	2299	0,03	49060,66	Oui
101	Construction	4	14/01/2003 00:00	Compaq Presario 100	11	1650	0	18150	Oui
117	Pharmaceutique	5	14/03/2005 00:00	IBM 500	8	2299	0,015	18116,12	Oui
121	Distribution	6	11/09/2009 00:00	AST Intel 200	23	3190	0,03	71168,9	Oui
102	Machines/Outil	7	08/07/2003 00:00	Compaq Presario 100	24	1650	0	39600	Oui
122	Machines/Outil	8	24/07/2006 00:00	IBM 500	23	2299	0	52877	Oui
104	Construction	9	26/06/2004 00:00	AST Intel 150	4	2690	0,03	10437,2	Oui
107	Distribution	10	15/10/2009 00:00	Compaq Presario 100	24	1650	0,03	38412	Oui
115	Éducation	11	03/08/2008 00:00	AST Intel 200	25	3190	0	79750	Oui
116	Éducation	12	18/04/2003 00:00	Compaq Presario 100	17	1650	0,015	27629,25	Oui
109	Construction	13	24/02/2001 00:00	AST Intel 200	18	3190	0,03	55697,4	Oui
117	Construction	14	02/09/2005 00:00	Compaq Presario 100	9	1650	0	14850	Oui
123	Pharmaceutique	15	09/03/2005 00:00	IBM 500	22	2299	0,015	49819,33	Oui
116	Machines/Outil	16	11/05/2000 00:00	AST Intel 150	19	2690	0,015	50343,35	Oui
111	Construction	17	27/03/2003 00:00	Compaq Presario 100	23	1650	0,03	36811,5	Oui
111	Assurances	18	02/03/2002 00:00	IBM 500	14	2299	0,03	31220,42	Oui
103	Construction	19	16/03/2001 00:00	AST Intel 200	14	3190	0,03	43320,2	Oui
121	Pharmaceutique	20	11/12/2001 00:00	AST Intel 200	4	3190	0	12760	Non
118	Éducation	21	21/10/2006 00:00	AST Intel 200	5	3190	0	15950	Oui
120	Banques	22	22/01/2000 00:00	IBM 500	12	2299	0,015	27174,18	Oui
117	Banques	23	06/06/2004 00:00	AST Intel 150	5	2690	0	13450	Oui
101	Construction	24	18/08/2001 00:00	Compaq Presario 100	13	1650	0,015	21128,25	Oui
103	Assurances	25	15/03/2009 00:00	AST Intel 150	12	2690	0,015	31795,8	Oui
101	Construction	26	17/04/2008 00:00	IBM 500	5	2299	0	11495	Oui
121	Pharmaceutique	27	19/11/2000 00:00	AST Intel 200	22	3190	0,03	68074,6	Oui
108	Pharmaceutique	28	15/01/2000 00:00	AST Intel 200	13	3190	0,015	40847,95	Oui
103	Assurances	29	15/10/2006 00:00	Compaq Presario 100	8	1650	0,015	13002	Non

Et écrivez le code Office Script suivant non-optimisé:



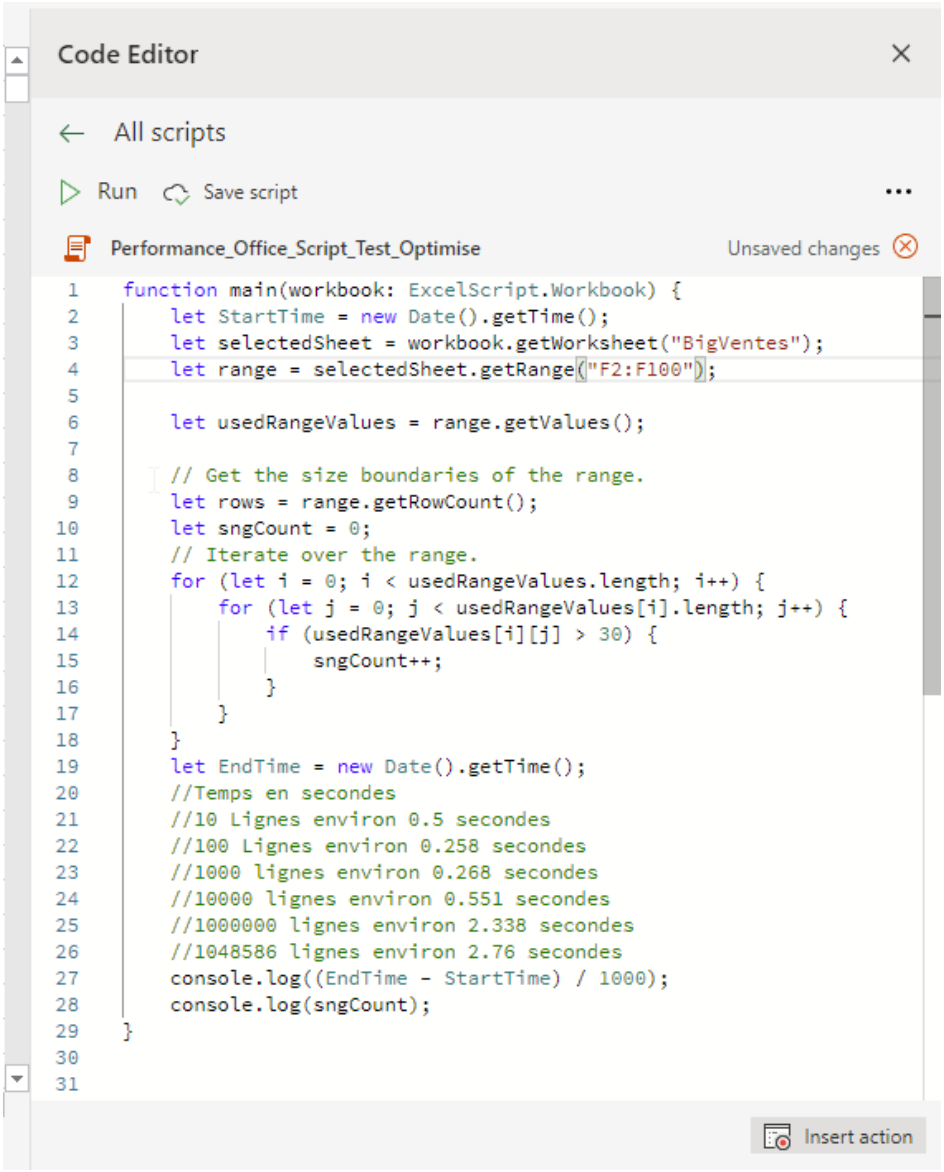
Si nous exécutons, nous avons alors comme indiqué dans les commentaires:



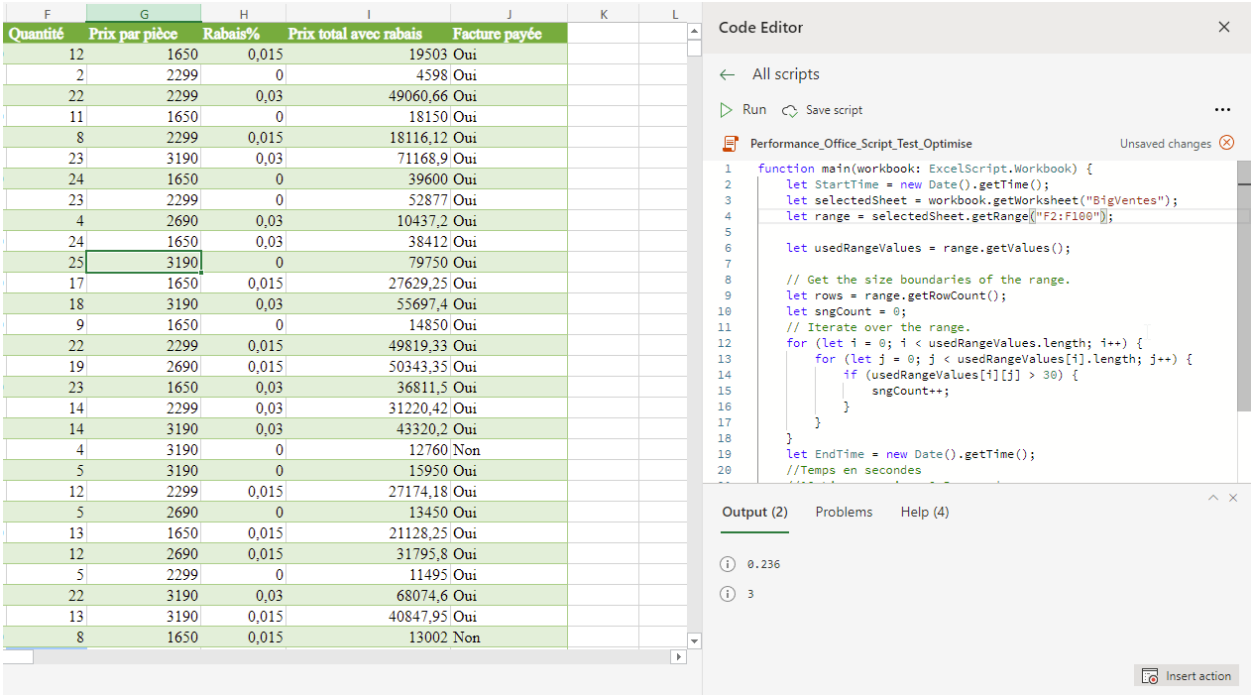
	Quantité	Prix par pièce	Rabais%	Prix total avec rabais	Facture payée
0	12	1650	0,015	19503	Oui
	2	2299	0	4598	Oui
	22	2299	0,03	49060,66	Oui
0	11	1650	0	18150	Oui
	8	2299	0,015	18116,12	Oui
	23	3190	0,03	71168,9	Oui
0	24	1650	0	39600	Oui
	23	2299	0	52877	Oui
	4	2690	0,03	10437,2	Oui
0	24	1650	0,03	38412	Oui
	25	3190	0	79750	Oui
0	17	1650	0,015	27629,25	Oui
	18	3190	0,03	55697,4	Oui
0	9	1650	0	14850	Oui
	22	2299	0,015	49819,33	Oui
	19	2690	0,015	50343,35	Oui
0	23	1650	0,03	36811,5	Oui
	14	2299	0,03	31220,42	Oui
	14	3190	0,03	43320,2	Oui
	4	3190	0	12760	Non
	5	3190	0	15950	Oui
	12	2299	0,015	27174,18	Oui
	5	2690	0	13450	Oui
0	13	1650	0,015	21128,25	Oui
	12	2690	0,015	31795,8	Oui
	5	2299	0	11495	Oui
	22	3190	0,03	68074,6	Oui
	13	3190	0,015	40847,95	Oui
0	8	1650	0,015	13002	Non

Mais maintenant, écrivons une version optimisée:

Public
Internal



Ce qui donne quand exécuté:



Quantité	Prix par pièce	Rabais%	Prix total avec rabais	Facture payée
12	1650	0,015	19503	Oui
2	2299	0	4598	Oui
22	2299	0,03	49060,66	Oui
11	1650	0	18150	Oui
8	2299	0,015	18116,12	Oui
23	3190	0,03	71168,9	Oui
24	1650	0	39600	Oui
23	2299	0	52877	Oui
4	2690	0,03	10437,2	Oui
24	1650	0,03	38412	Oui
25	3190	0	79750	Oui
17	1650	0,015	27629,25	Oui
18	3190	0,03	55697,4	Oui
9	1650	0	14850	Oui
22	2299	0,015	49819,33	Oui
19	2690	0,015	50343,35	Oui
23	1650	0,03	36811,5	Oui
14	2299	0,03	31220,42	Oui
14	3190	0,03	43320,2	Oui
4	3190	0	12760	Non
5	3190	0	15950	Oui
12	2299	0,015	27174,18	Oui
5	2690	0	13450	Oui
13	1650	0,015	21128,25	Oui
12	2690	0,015	31795,8	Oui
5	2299	0	11495	Oui
22	3190	0,03	68074,6	Oui
13	3190	0,015	40847,95	Oui
8	1650	0,015	13002	Non

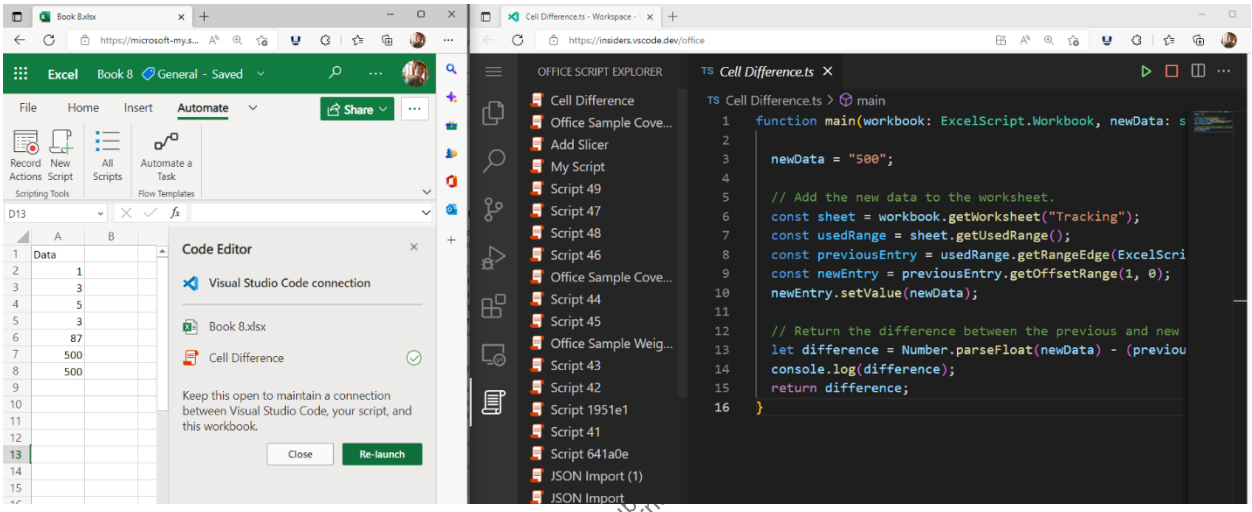
Public
Internal

13 Édition dans Visual Studio Code

En 2022 Microsoft avait activé une option pour ouvrir dans les fichiers Office Script dans la version web de Visual Studio Code:

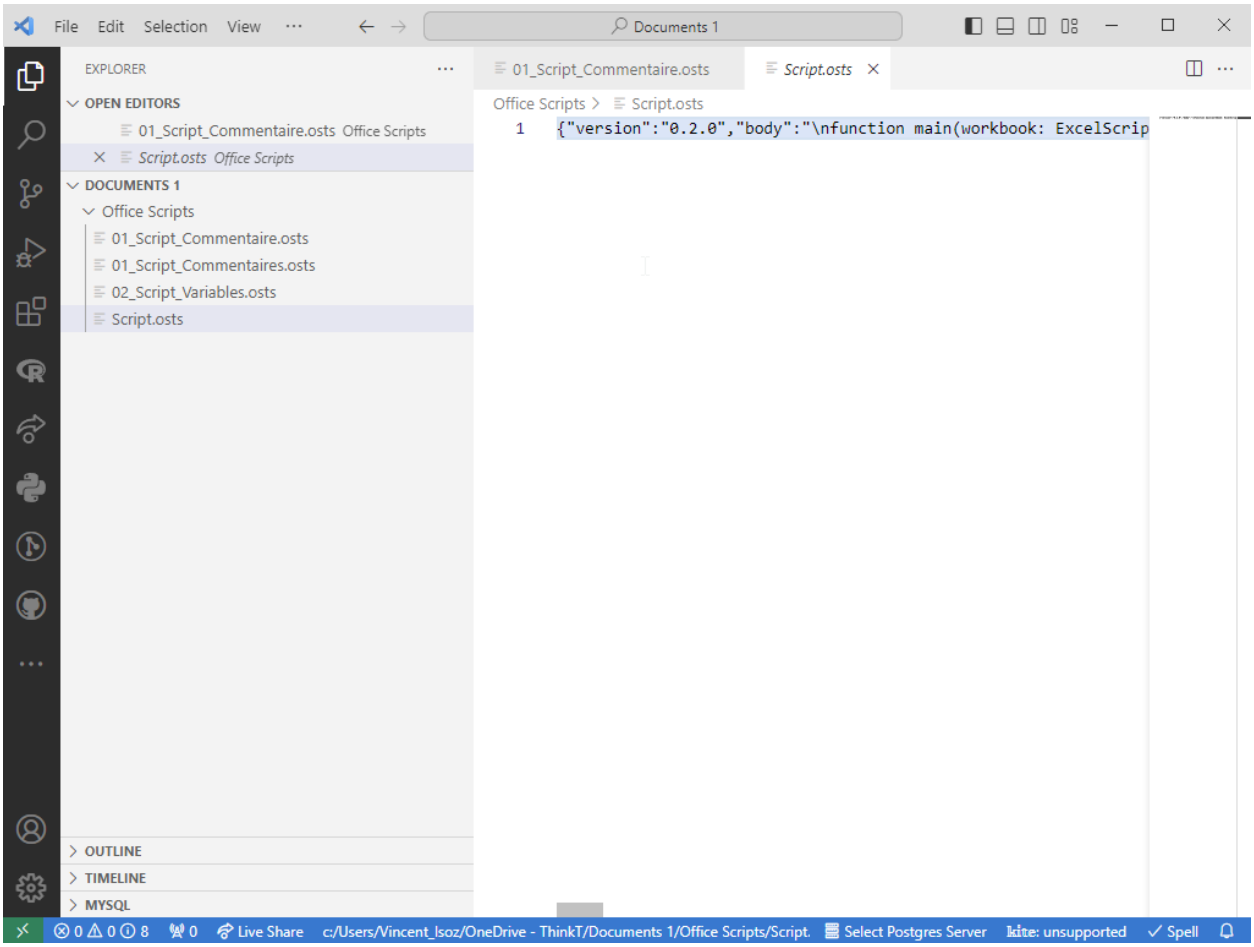
https://vscode.dev/

mais l'ont retiré après quelques semaines (le 14 Juin 2023):

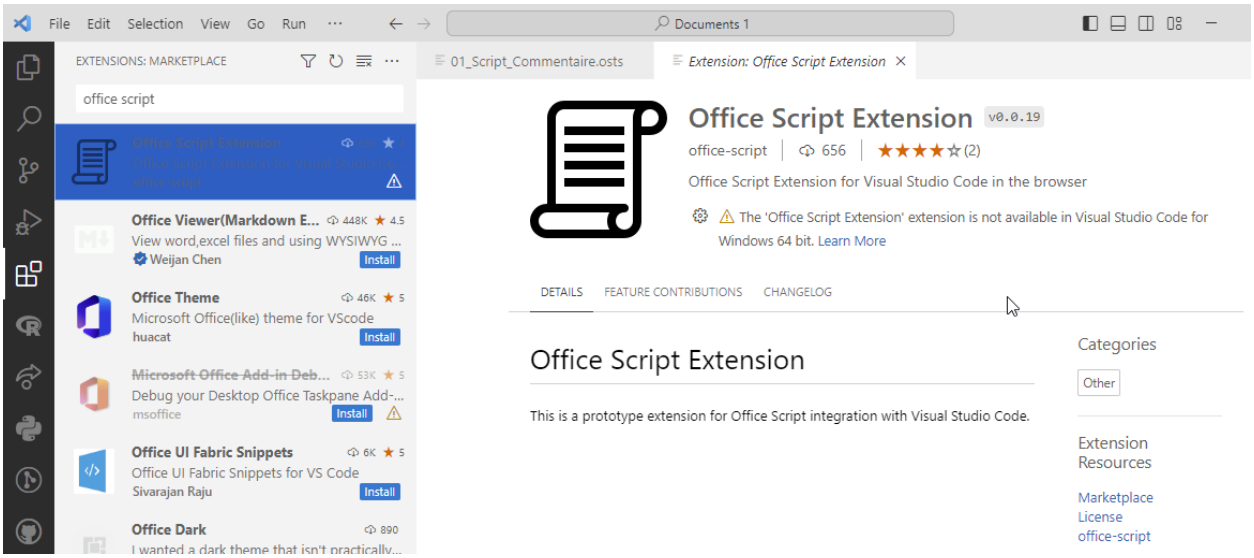


13.1 Visual Studio Code Desktop

Depuis on ne peut plus rien faire et ouvrir un fichier Office Script dans Visual Studio Code Desktop ressemble à ceci:



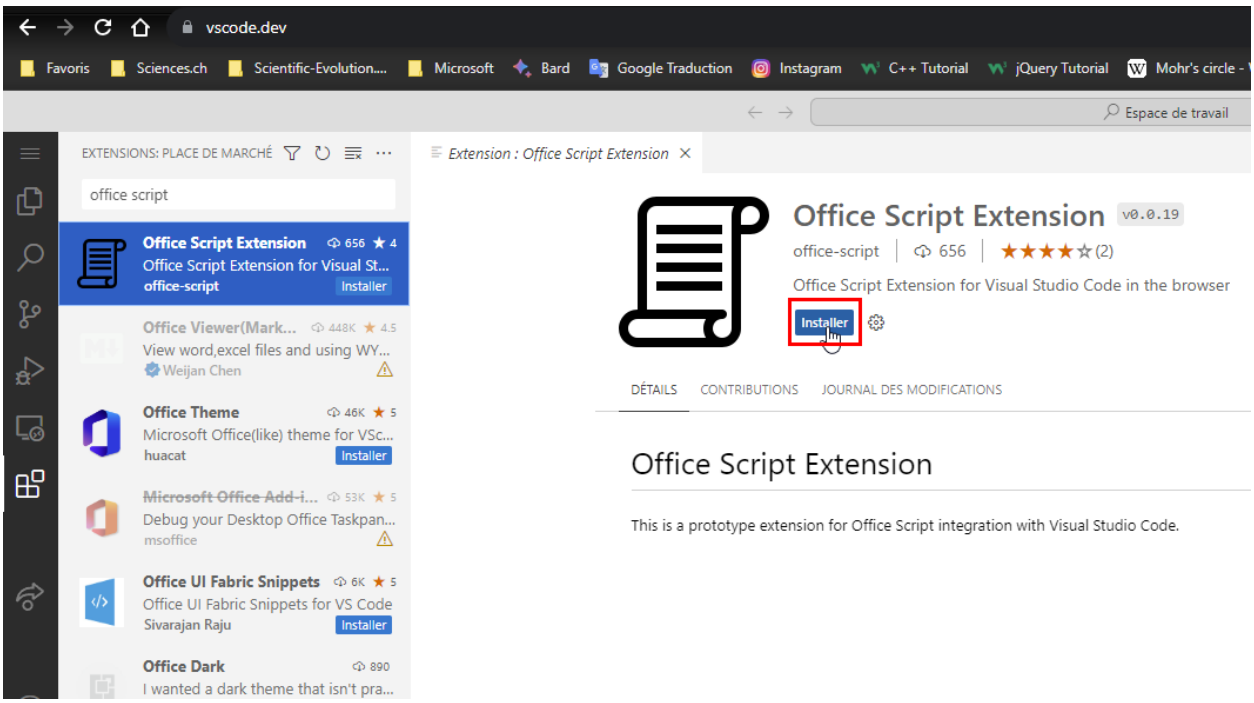
Et la seule extension disponible pour Office Script n'est pas installable (car elle est réservée à Visual Studio Code pour le web) et renvoie vers un lien mort:



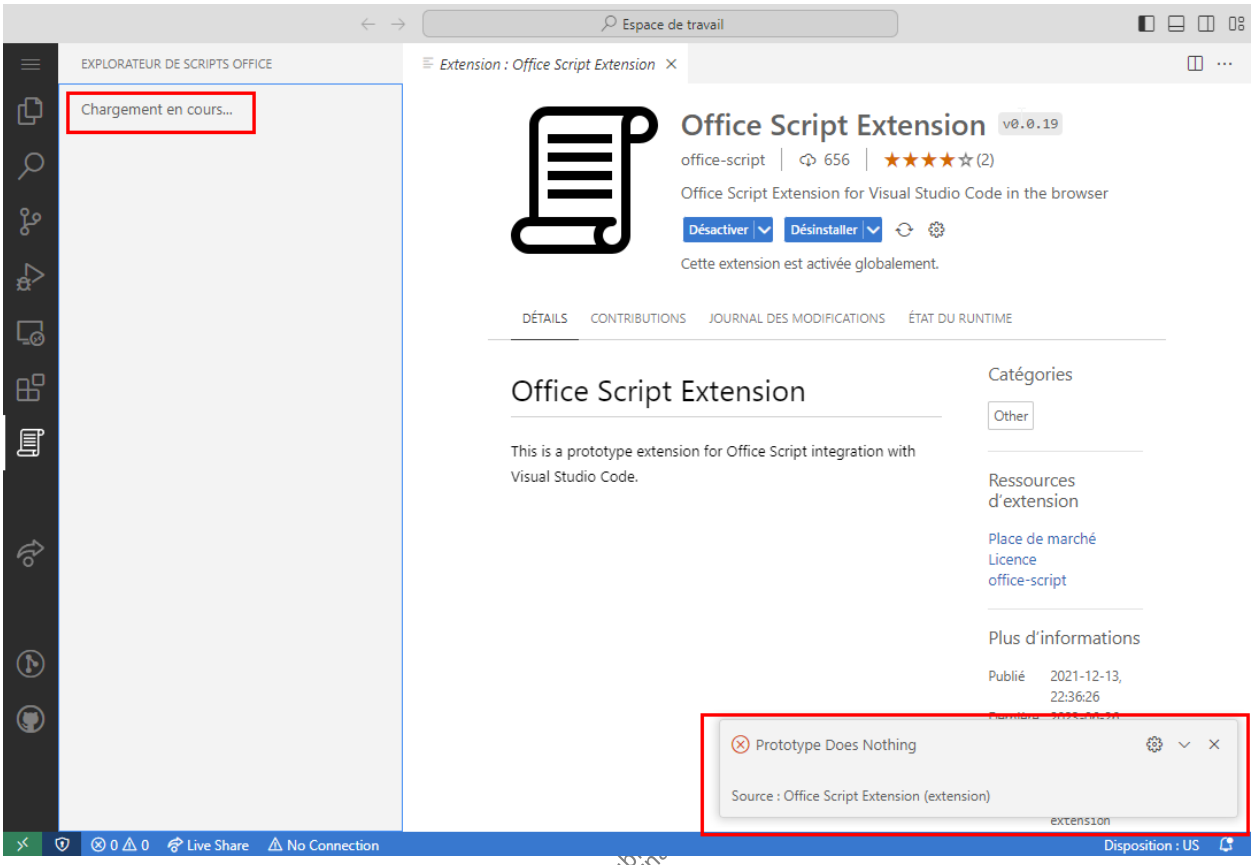
13.2 Visual Studio Code Web

Visual Studio Code Web a une extension pour Office Script:

https://vscode.dev/

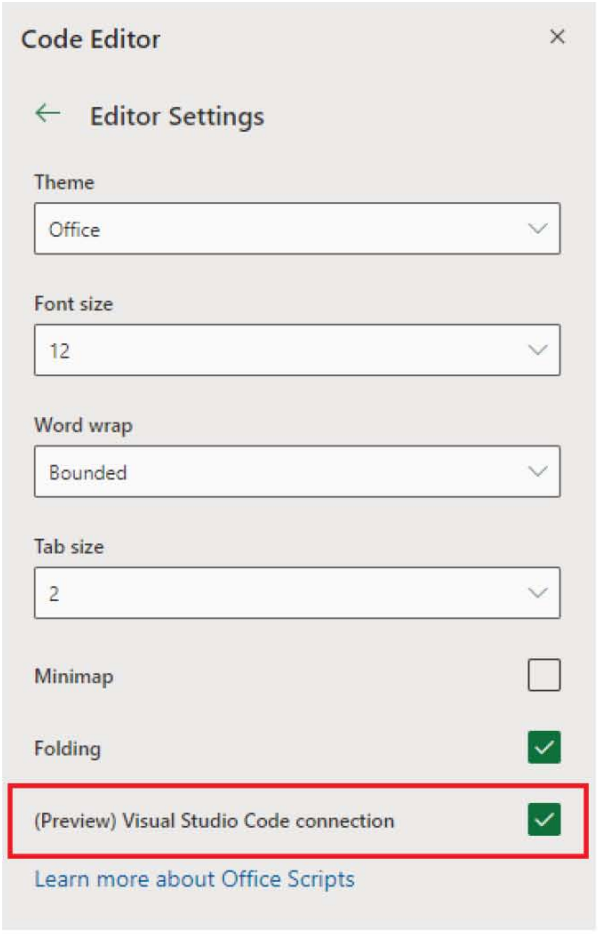


Mais si on essaie de l'installer on déchante… :

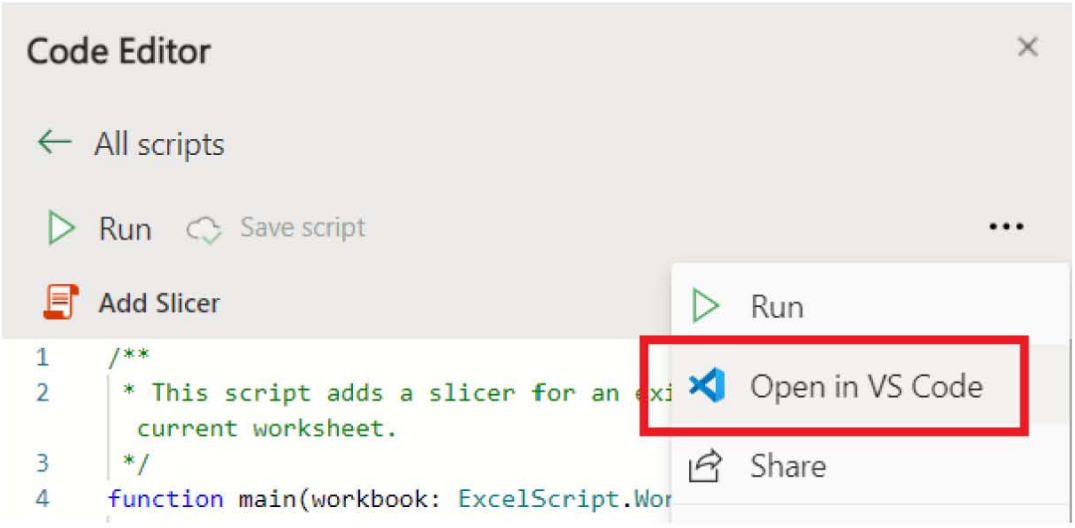


Pub. Interne

Il faut savoir que a une époque nous avions une option pour ouvrir Office Script dans Visual Code comme le prouve la capture d'écran ci-dessous:



Ce qui donnait:



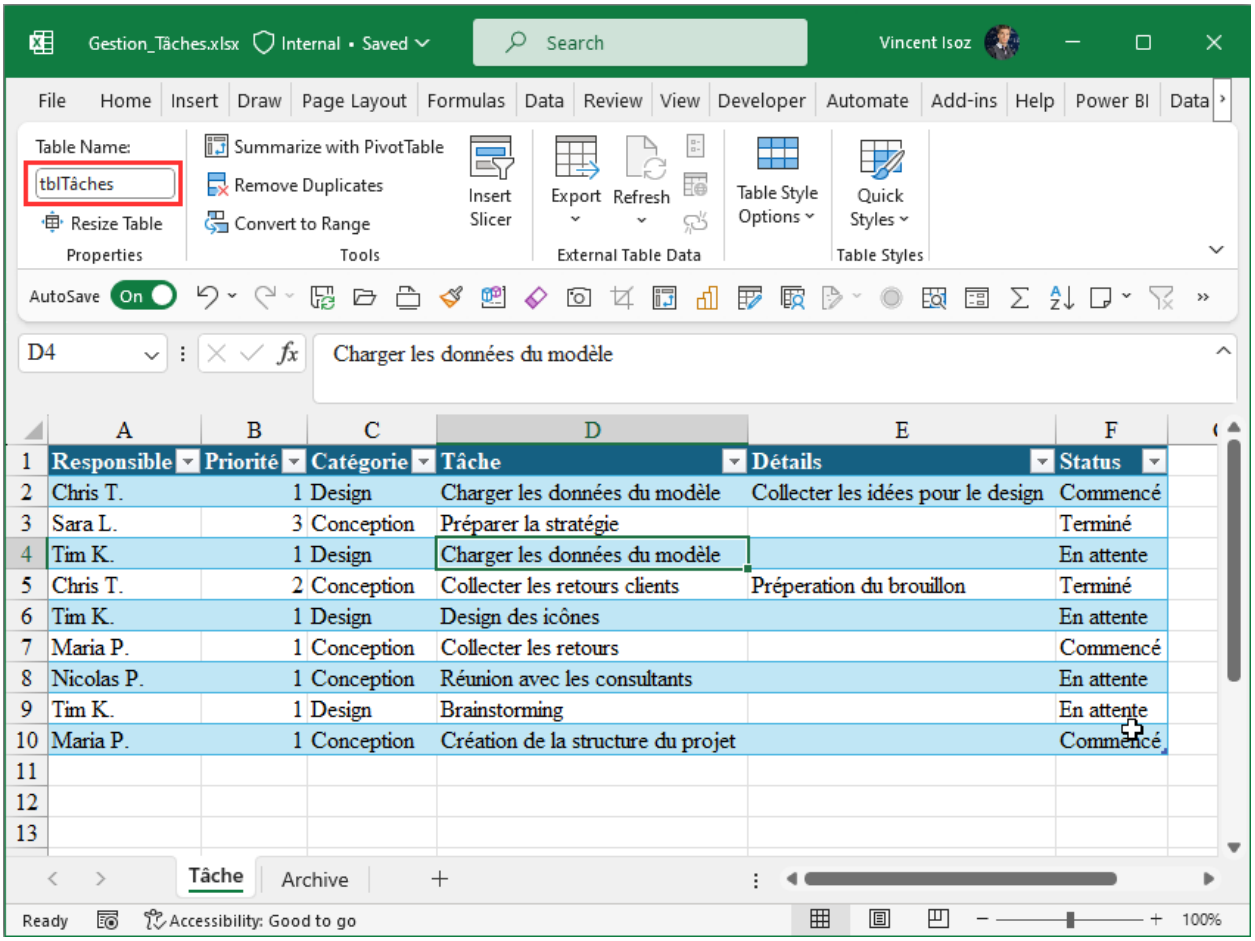
Mais cette option a été retirée entre-temps.

14 Exemples d'association avec Power Automate

Comme nous l'avons déjà mentionné au début, de nombreuses choses ne peuvent pas être faites sans Power Automate. Nous supposerons que le lecteur aura déjà suivi un cours ou étudié ce qu'est Power Automate et nous allons ici nous focaliser sur des exemples classiques d'utilisation de ce dernier avec Office Script!

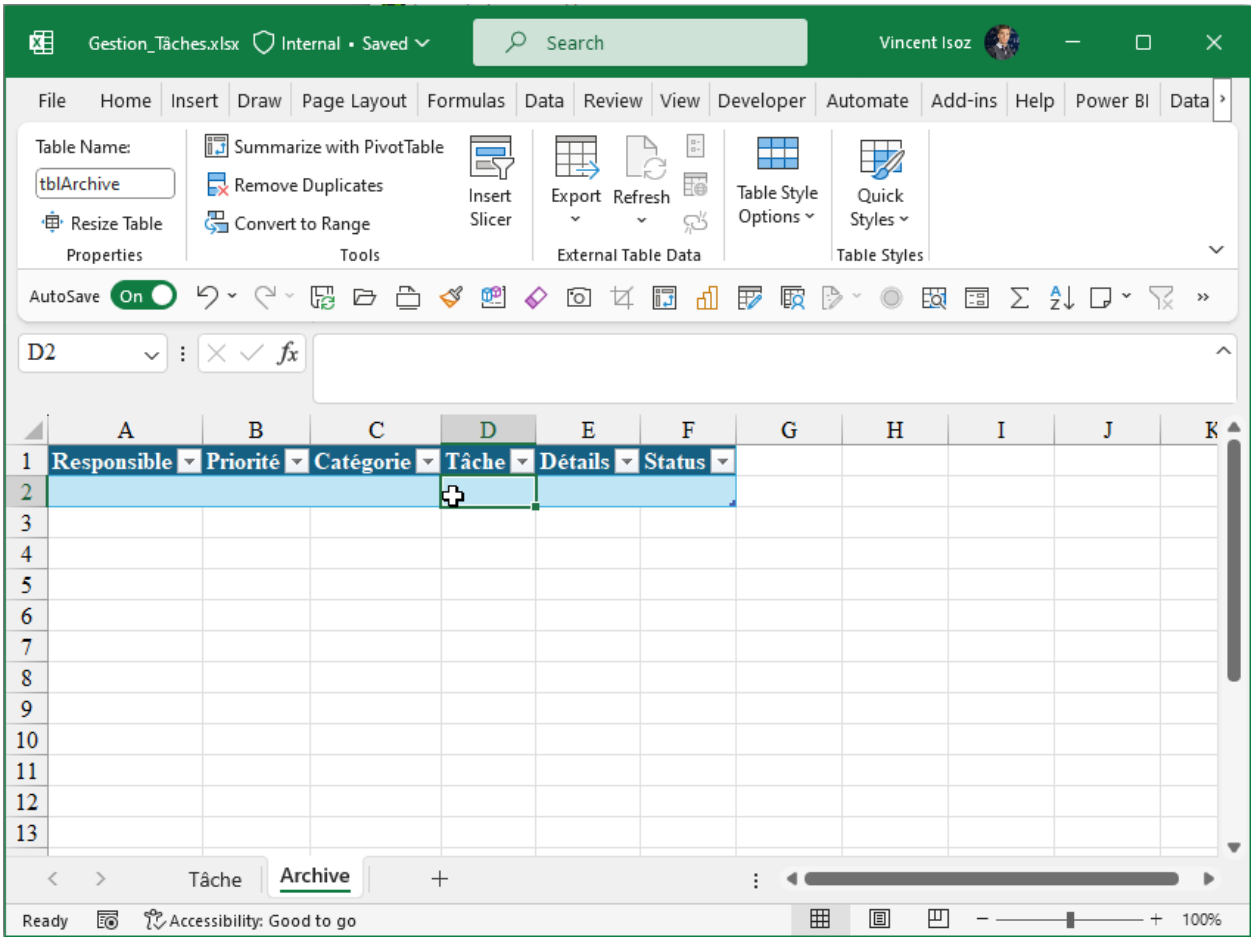
14.1 Copie de données d'une feuille Excel à une autre

Considérons le fichier avec la table *tblTâches* dans la feuille *Tâche*:



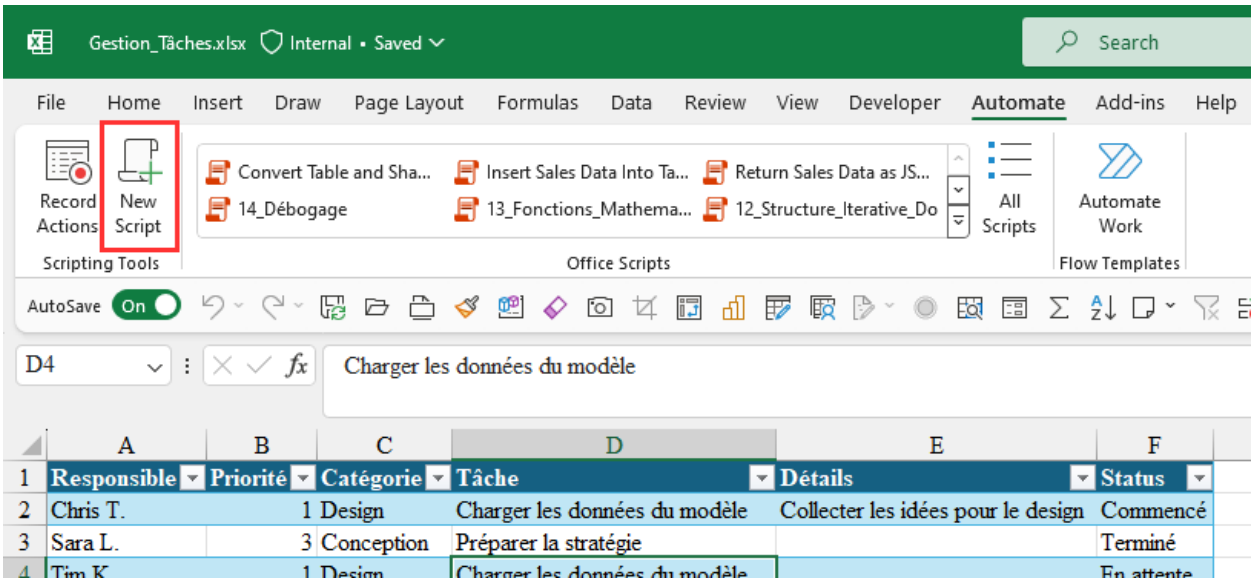
	A	B	C	D	E	F
1	Responsible	Priorité	Catégorie	Tâche	Détails	Status
2	Chris T.	1	Design	Charger les données du modèle	Collecter les idées pour le design	Commencé
3	Sara L.	3	Conception	Préparer la stratégie		Terminé
4	Tim K.	1	Design	Charger les données du modèle		En attente
5	Chris T.	2	Conception	Collecter les retours clients	Préperation du brouillon	Terminé
6	Tim K.	1	Design	Design des icônes		En attente
7	Maria P.	1	Conception	Collecter les retours		Commencé
8	Nicolas P.	1	Conception	Réunion avec les consultants		En attente
9	Tim K.	1	Design	Brainstorming		En attente
10	Maria P.	1	Conception	Création de la structure du projet		Commencé
11						
12						
13						

Et dans le même fichier, la feuille *Archive* avec la table *tblArchive*:

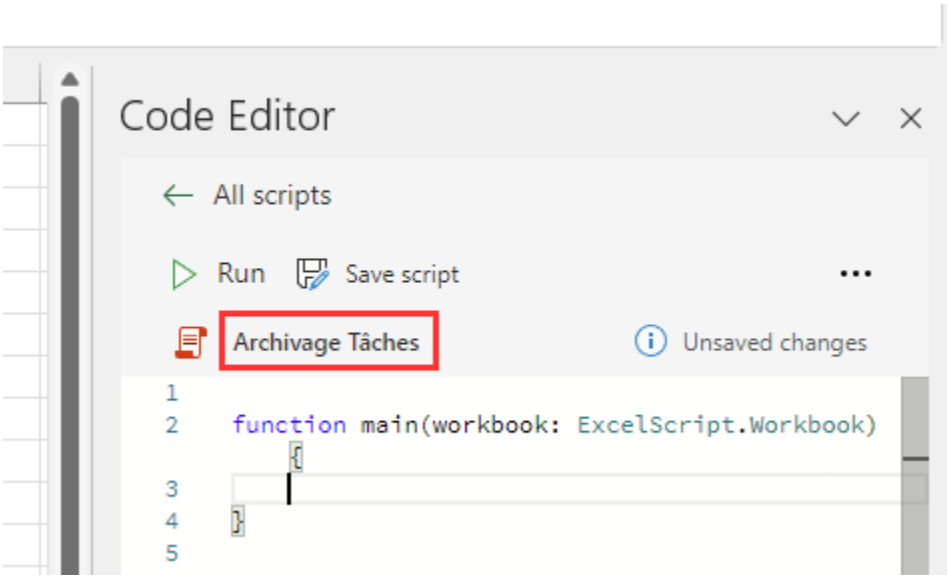


Ce que nous voulons c'est un script, qui peut être exécuté à temps d'intervalles régulier, qui copier les tâches en mode *Terminé*, dans la feuille *Archive*.

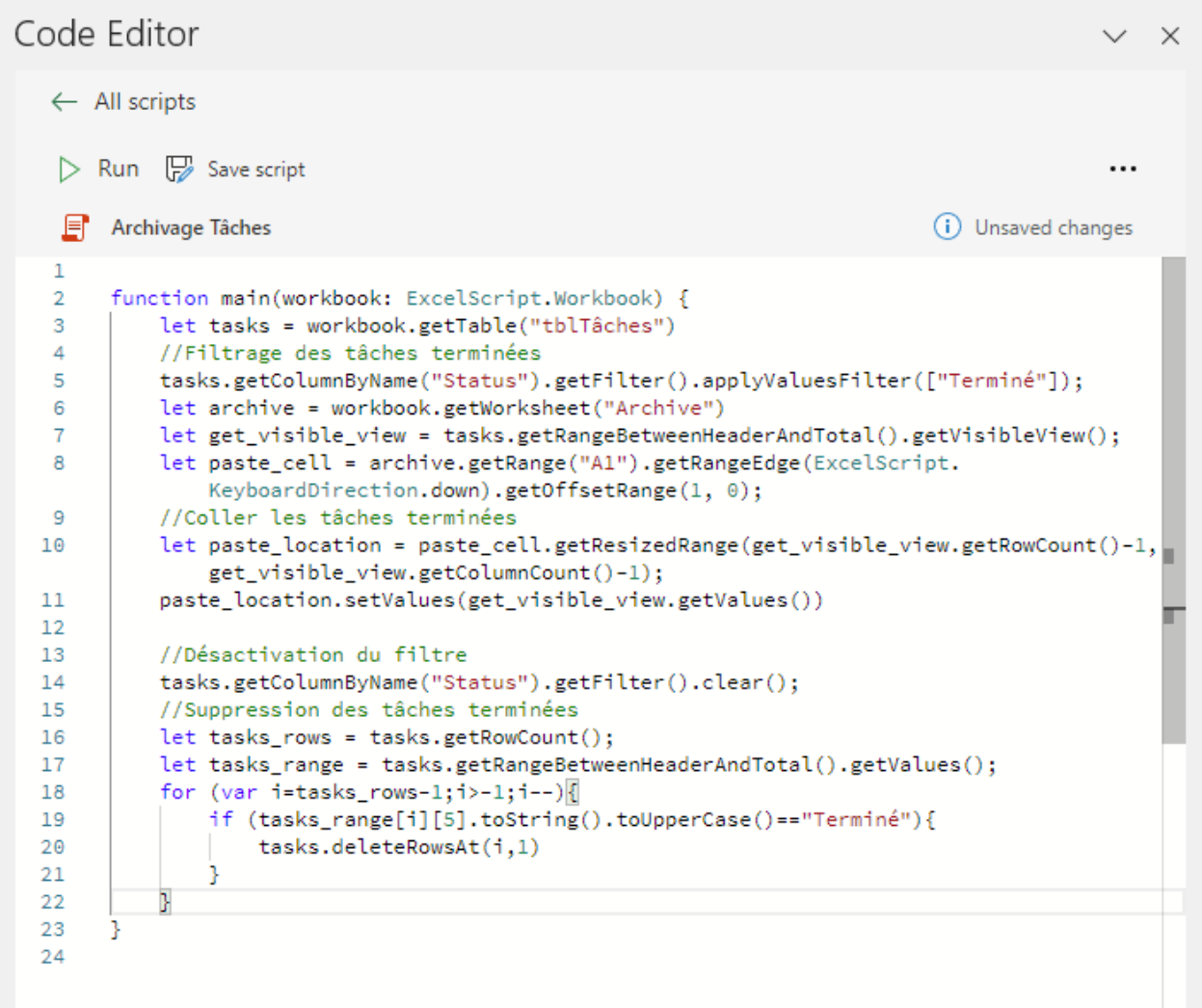
Pour cela, nous allons donc créer un script:



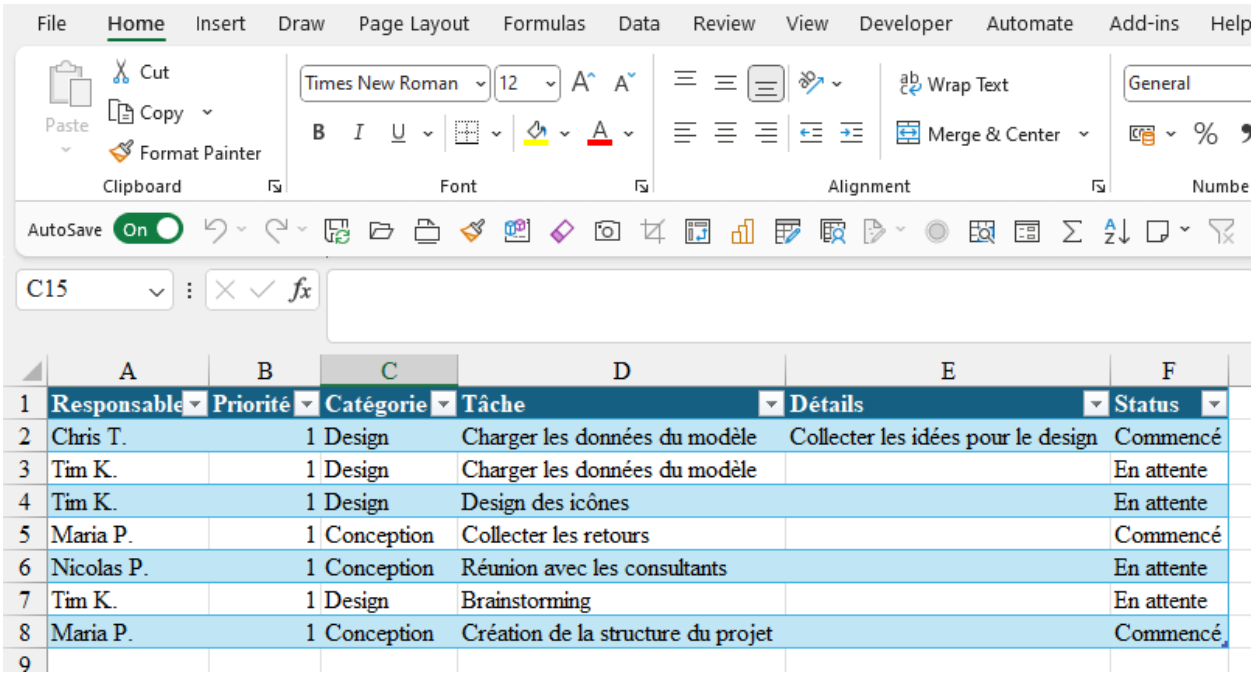
Script que nous allons de suite nommer:



Nous écrivons le code non-trivial suivant (on recommande les participants d'essayer de faire cela avec une macro OfficeScript enregistrée juste pour voir…):

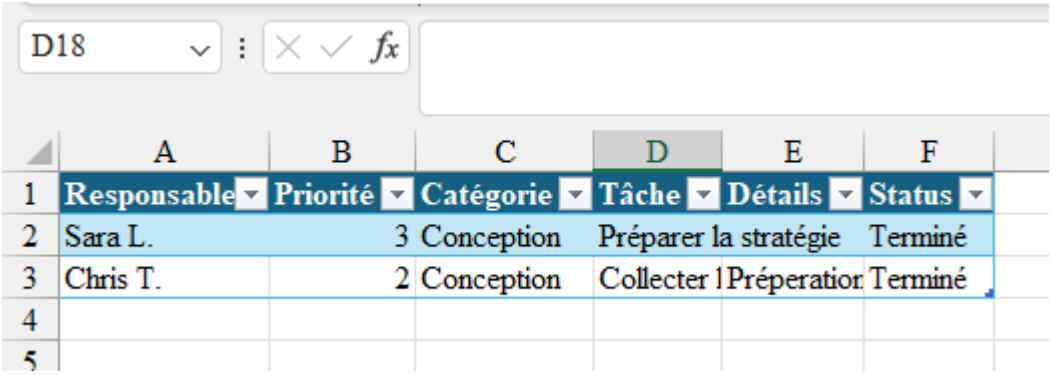


Si nous exécutons le script, nous obtenons alors:



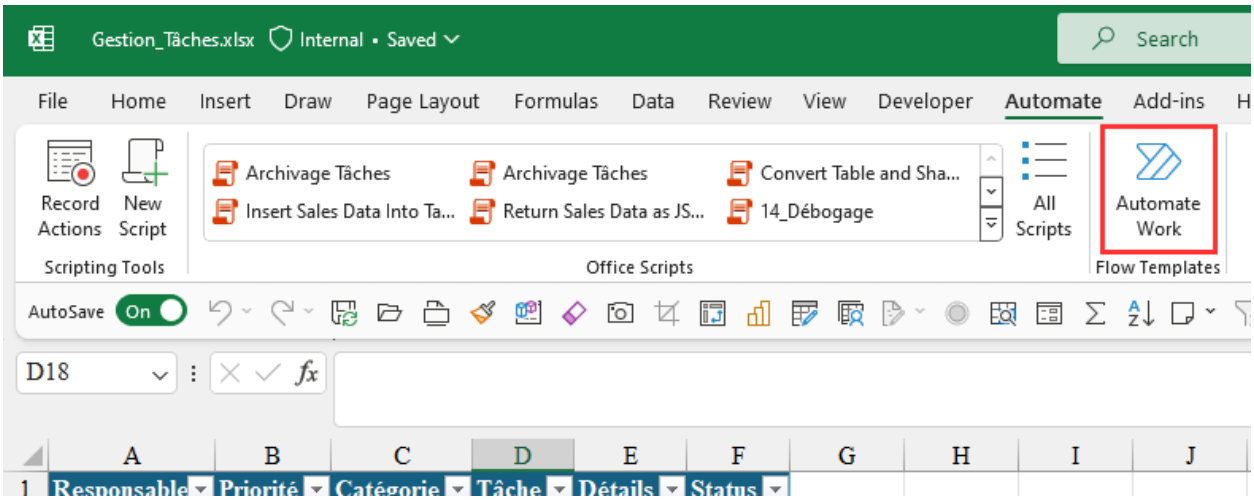
	A	B	C	D	E	F
1	Responsable	Priorité	Catégorie	Tâche	Détails	Status
2	Chris T.	1	Design	Charger les données du modèle	Collecter les idées pour le design	Commencé
3	Tim K.	1	Design	Charger les données du modèle		En attente
4	Tim K.	1	Design	Design des icônes		En attente
5	Maria P.	1	Conception	Collecter les retours		Commencé
6	Nicolas P.	1	Conception	Réunion avec les consultants		En attente
7	Tim K.	1	Design	Brainstorming		En attente
8	Maria P.	1	Conception	Création de la structure du projet		Commencé
9						

Et:

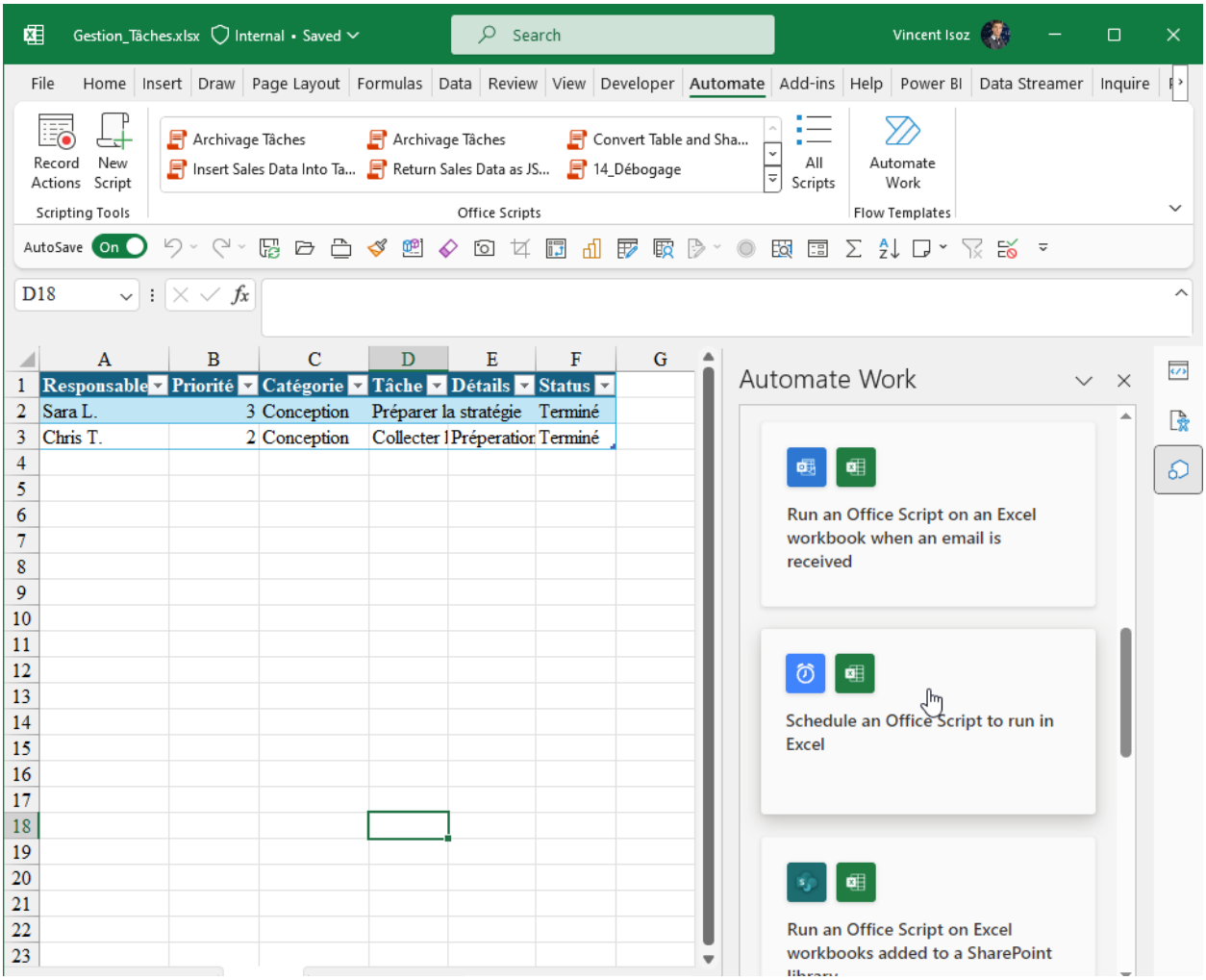


	A	B	C	D	E	F
1	Responsable	Priorité	Catégorie	Tâche	Détails	Status
2	Sara L.	3	Conception	Préparer la stratégie		Terminé
3	Chris T.	2	Conception	Collecter l	Préperation	Terminé
4						
5						

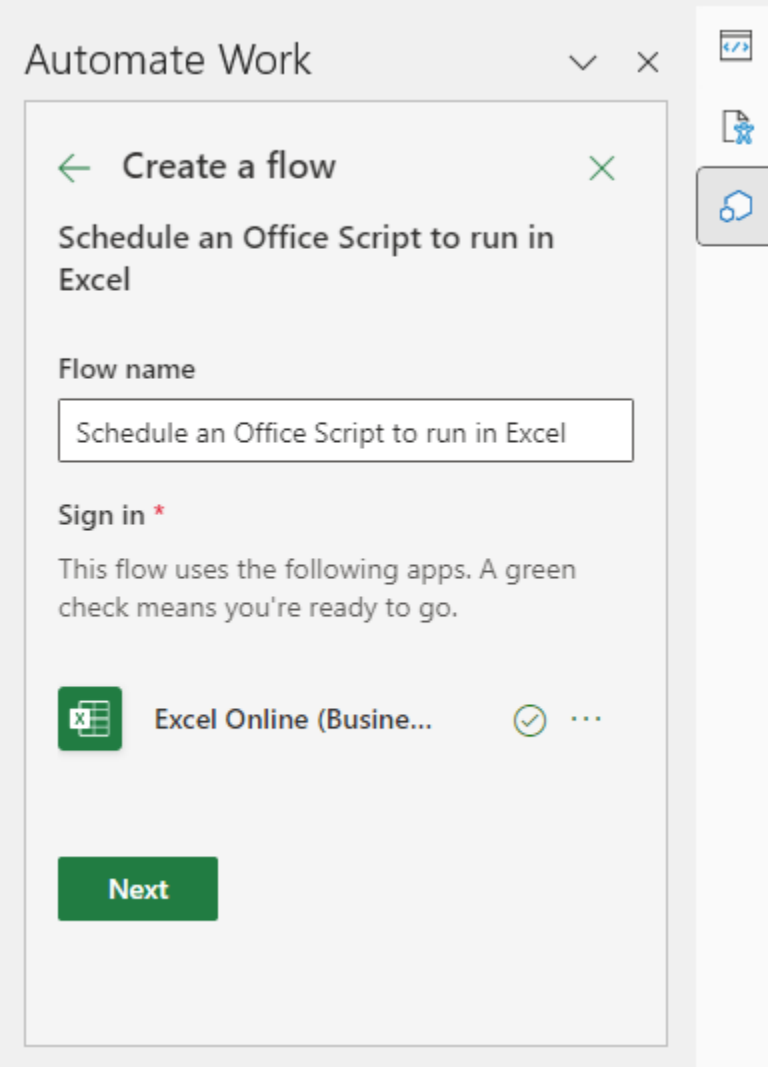
Maintenant nous voulons automatiser cette procédure périodiquement. Pour faire cela, nous cliquons sur le bouton **Automate Work**:



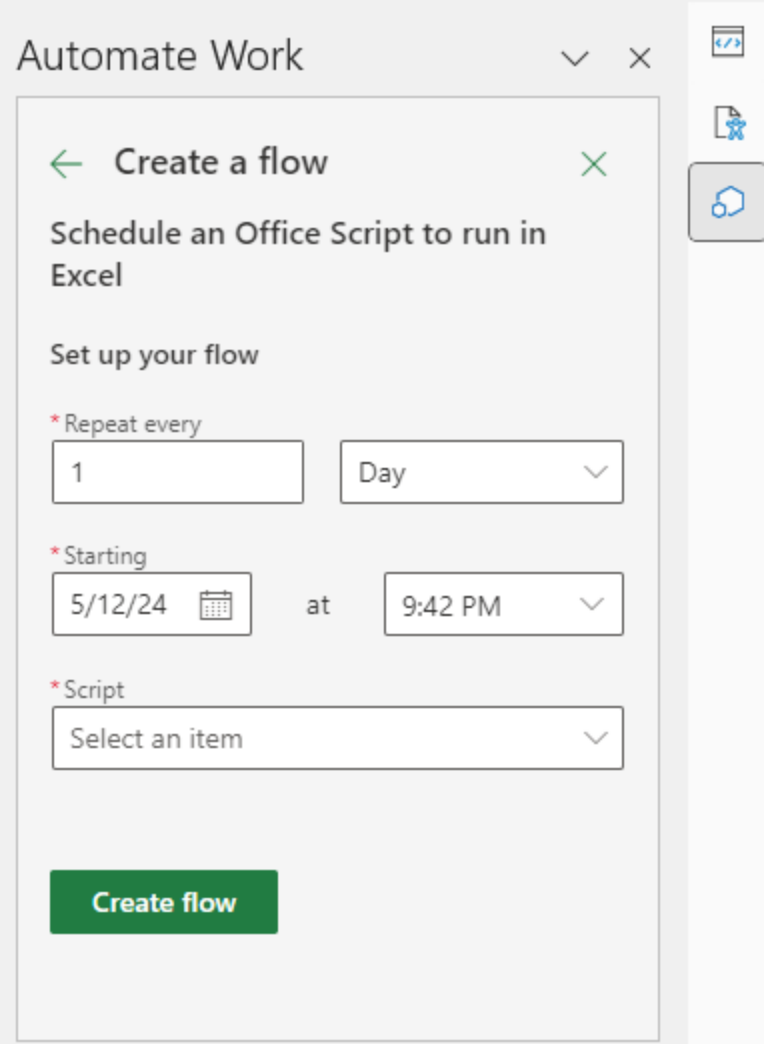
Et dans la panneau qui apparaît, nous prenons le modèle Schedule an *Office Script to run in Excel*:



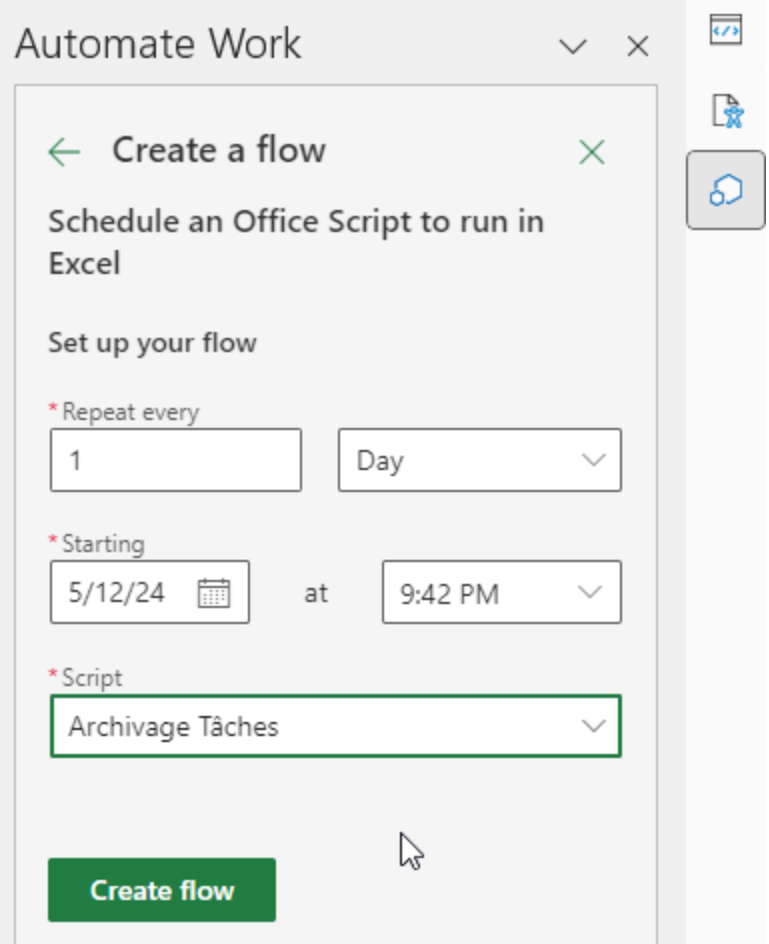
Nous avons alors l'assistant suivant qui apparaît:



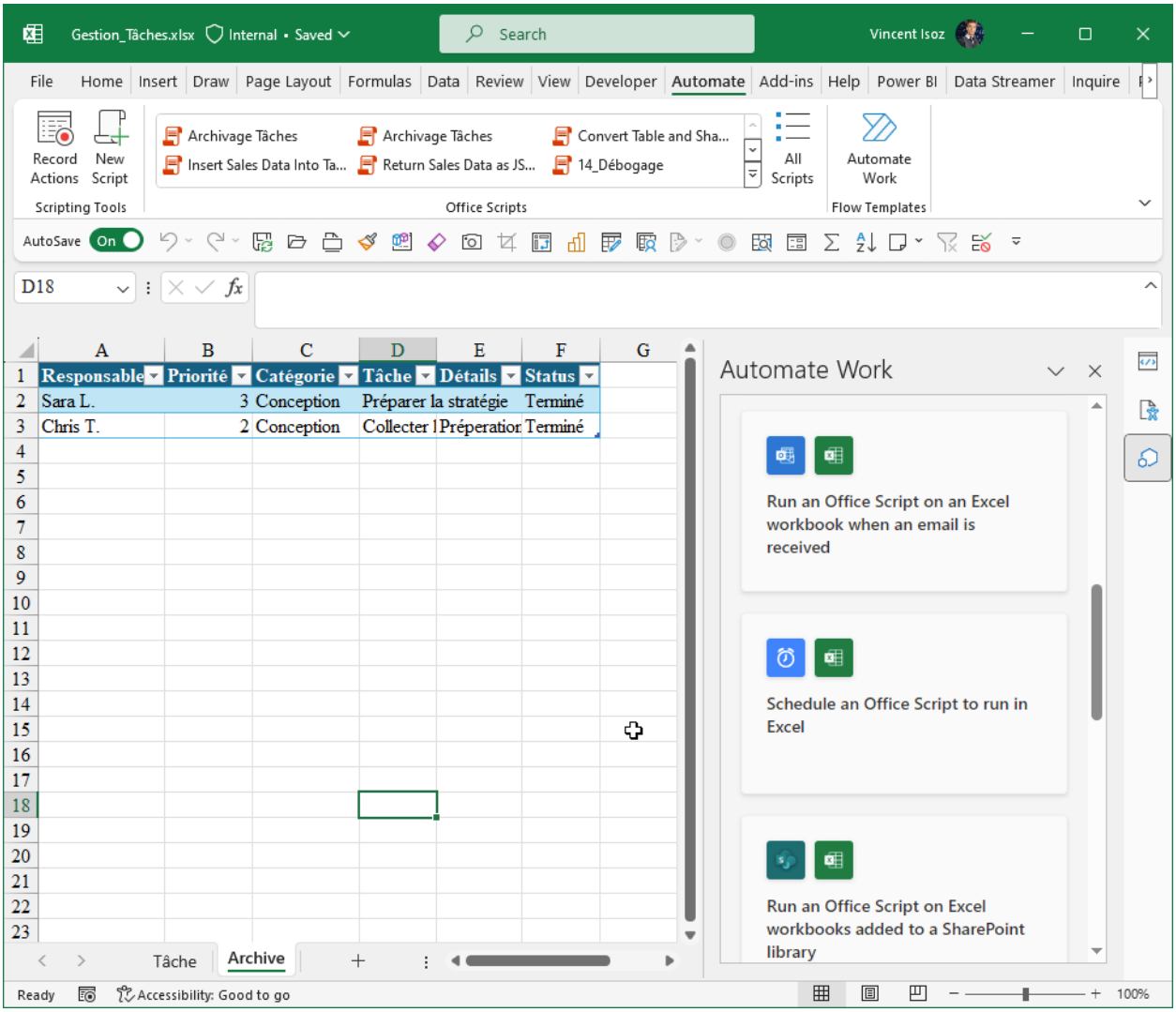
Nous cliquons sur **Next**:



Et dans *Select an item*, nous prenons:

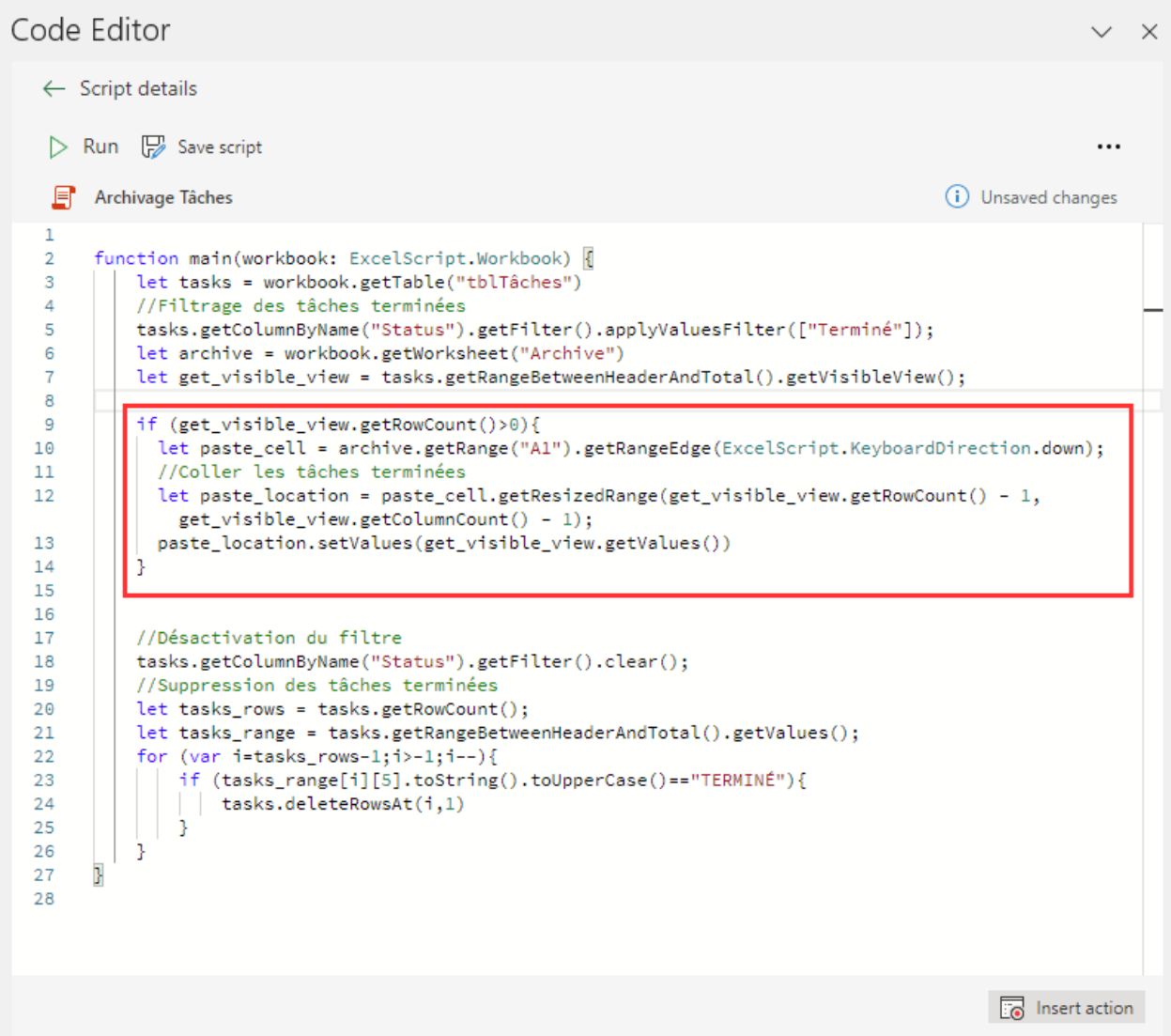


Et nous validons par **Create flow**:



Et voilà le Flow est créé! Il reste plus qu'à attendre qu'il s'exécute automatiquement.

Pour éviter les erreurs nous pouvons rajouter une condition au code:

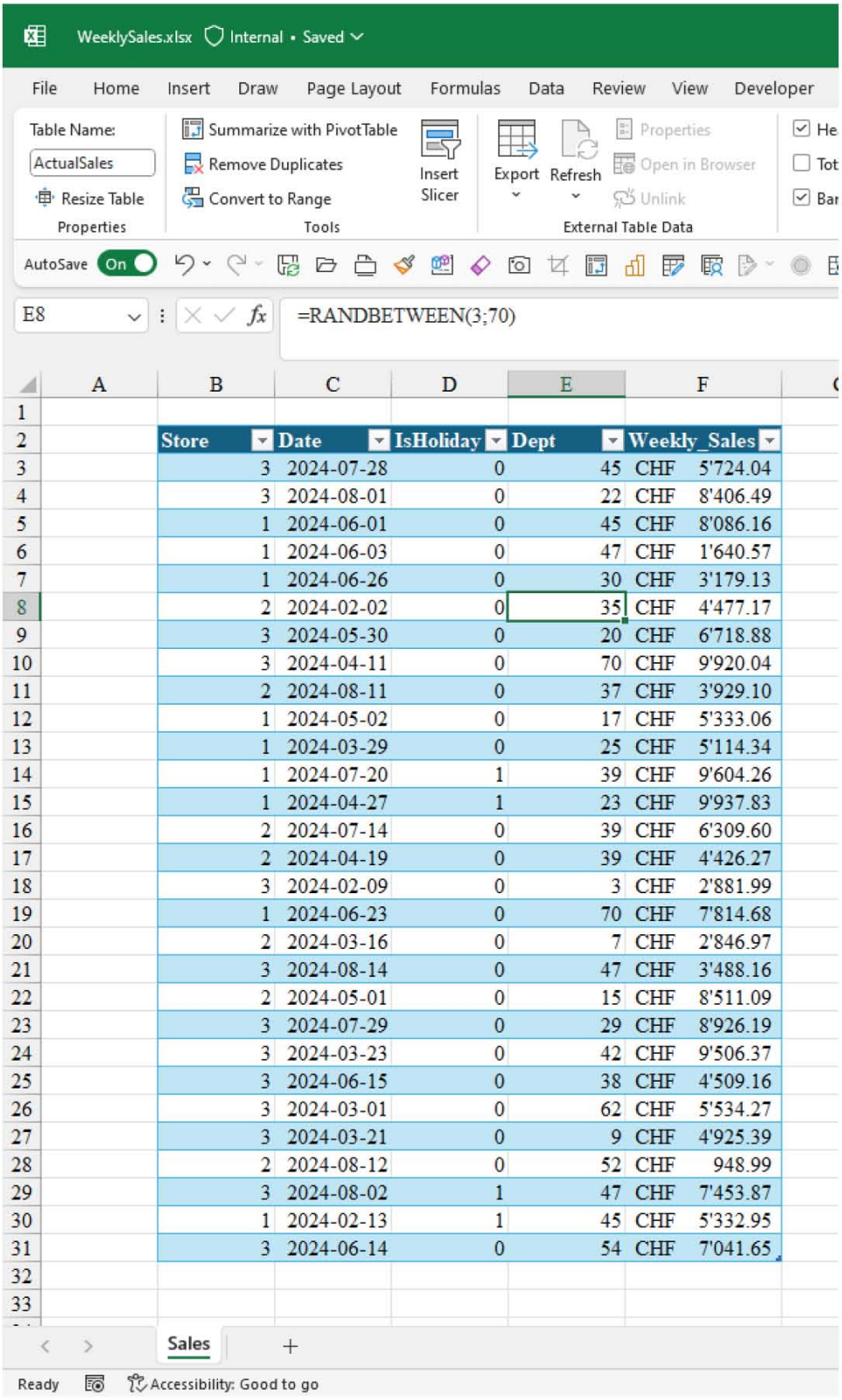


14.2 Copie de données d'un fichier XL à un autre et envoi par email des graphs et tables

Imaginez le scénario suivant : une chaîne de supermarchés enregistre toutes ses données de ventes dans des feuilles de calcul Excel (…); chaque lundi matin, un employé effectue le processus manuel et fastidieux suivant : il reçoit un e-mail contenant les données de ventes de la semaine précédente pour chaque magasin, il copie les données de la feuille de calcul dans une feuille de calcul principale contenant les données de ventes de l'année entière, exécute une Macro VBA qui actualise les tableaux croisés dynamiques et les graphiques et produit un rapport, ils envoient ensuite le rapport par e-mail à leur responsable.

L'exemple suivant illustre une solution qui automatise ce processus avec Office Script!

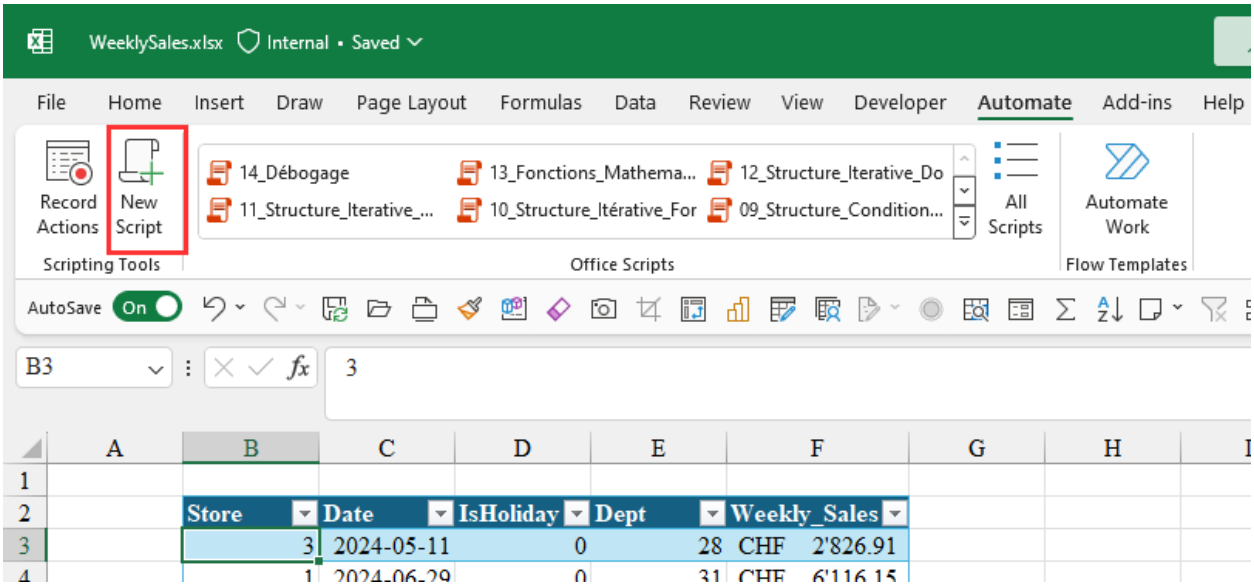
Voici un exemple de feuille de calcul de données de ventes hebdomadaires à partir de laquelle nous devons extraire des données:



	A	B	C	D	E	F
1						
2		Store	Date	IsHoliday	Dept	Weekly_Sales
3		3	2024-07-28	0	45	CHF 5'724.04
4		3	2024-08-01	0	22	CHF 8'406.49
5		1	2024-06-01	0	45	CHF 8'086.16
6		1	2024-06-03	0	47	CHF 1'640.57
7		1	2024-06-26	0	30	CHF 3'179.13
8		2	2024-02-02	0	35	CHF 4'477.17
9		3	2024-05-30	0	20	CHF 6'718.88
10		3	2024-04-11	0	70	CHF 9'920.04
11		2	2024-08-11	0	37	CHF 3'929.10
12		1	2024-05-02	0	17	CHF 5'333.06
13		1	2024-03-29	0	25	CHF 5'114.34
14		1	2024-07-20	1	39	CHF 9'604.26
15		1	2024-04-27	1	23	CHF 9'937.83
16		2	2024-07-14	0	39	CHF 6'309.60
17		2	2024-04-19	0	39	CHF 4'426.27
18		3	2024-02-09	0	3	CHF 2'881.99
19		1	2024-06-23	0	70	CHF 7'814.68
20		2	2024-03-16	0	7	CHF 2'846.97
21		3	2024-08-14	0	47	CHF 3'488.16
22		2	2024-05-01	0	15	CHF 8'511.09
23		3	2024-07-29	0	29	CHF 8'926.19
24		3	2024-03-23	0	42	CHF 9'506.37
25		3	2024-06-15	0	38	CHF 4'509.16
26		3	2024-03-01	0	62	CHF 5'534.27
27		3	2024-03-21	0	9	CHF 4'925.39
28		2	2024-08-12	0	52	CHF 948.99
29		3	2024-08-02	1	47	CHF 7'453.87
30		1	2024-02-13	1	45	CHF 5'332.95
31		3	2024-06-14	0	54	CHF 7'041.65
32						
33						

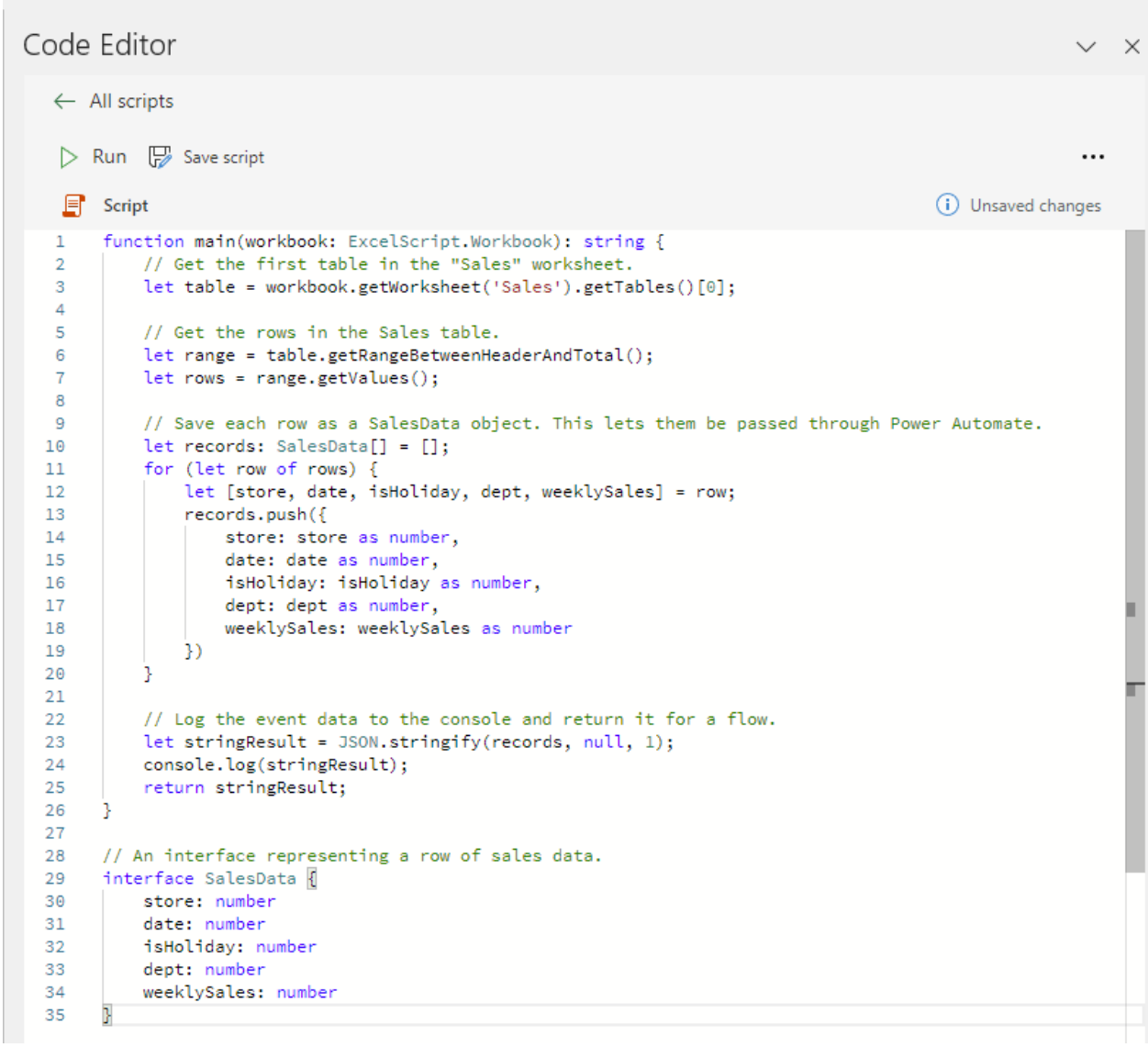
Nous avons besoin d'un script Office qui automatise le processus de sélection et de copie des données de vente à partir de la feuille de calcul qui arrive par courrier électronique:

Nous allons donc dans l'onglet **Automate** et prenons **New Script**:

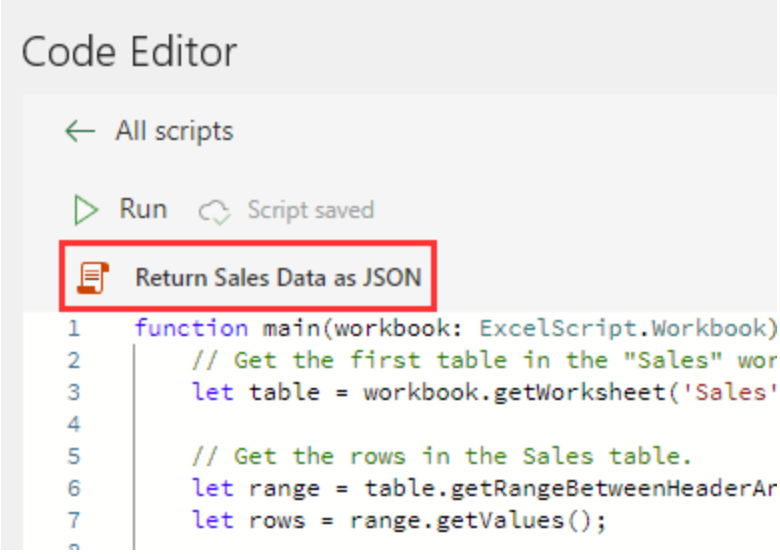


Et nous écrivons le code suivant:

Public
Internal

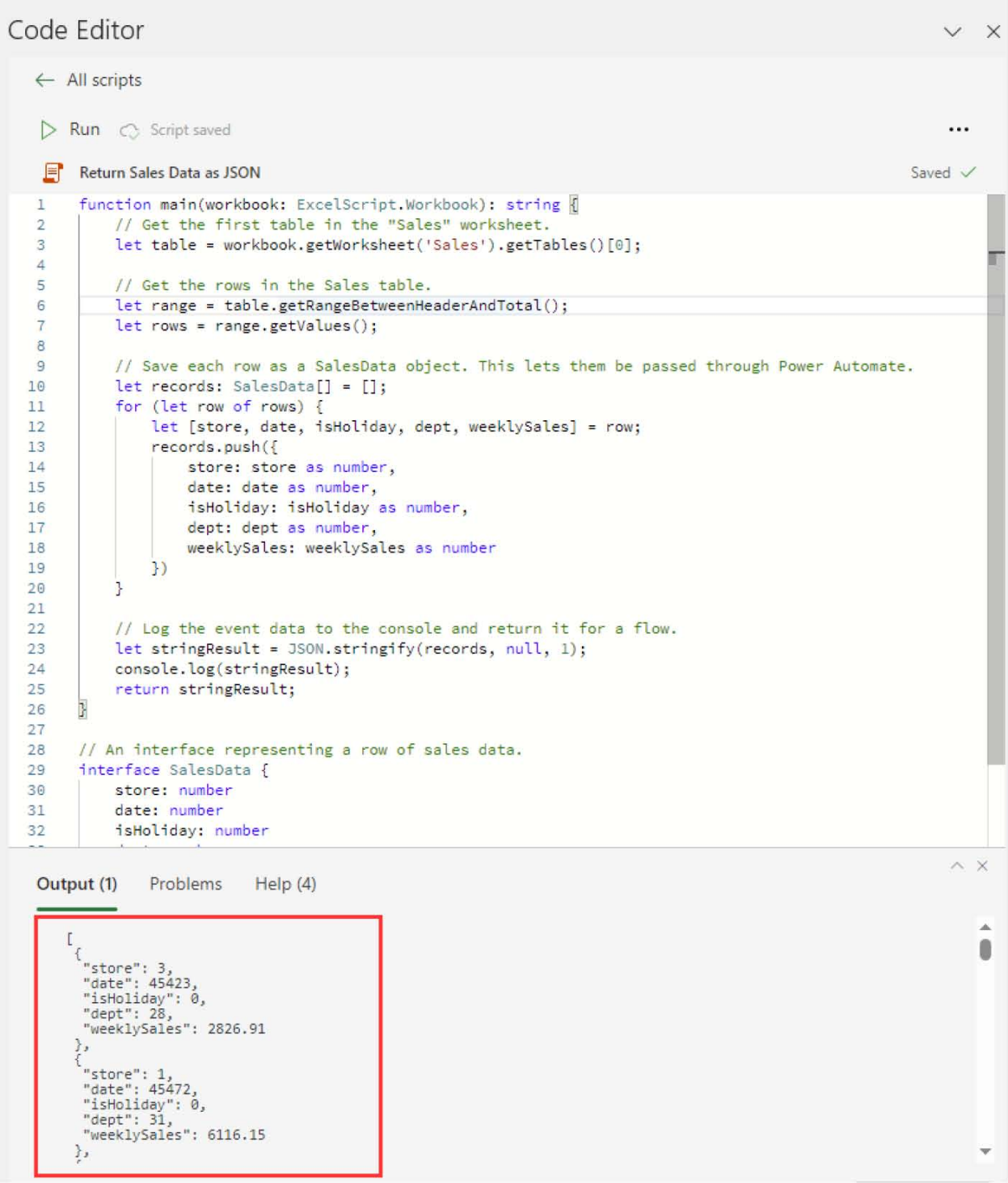


Script que nous allons renommer:



Si nous exécutons ce script, nous voyons bien la sortie JSON dans la console d'Output:

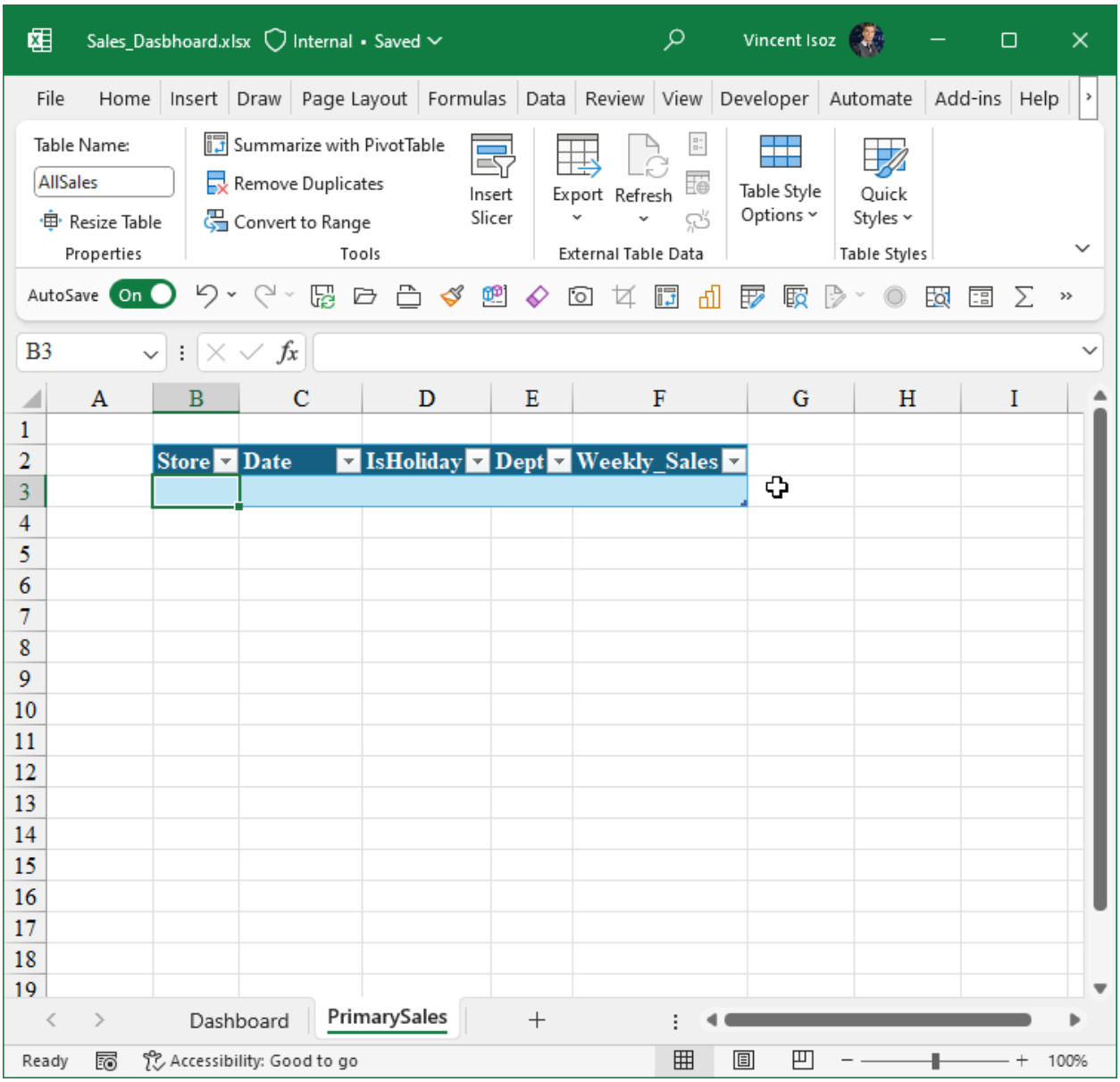
Public
Internal



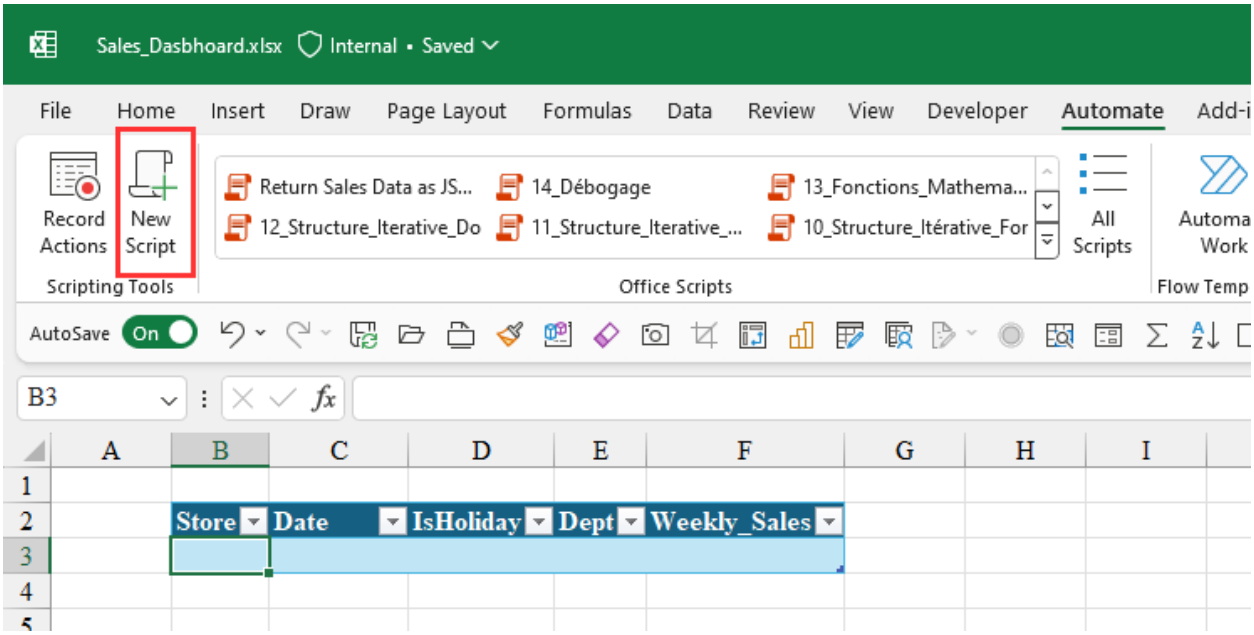
Le script Office ci-dessus extrait les données de la feuille de calcul *Sales* et les renvoie sous forme de chaîne JSON, ce qui est nécessaire pour qu'elles soient transmises en sortie dans un flux Power Automate. Voyons ce qu'il fait étape par étape. Tout d'abord, nous extrayons les données de la première table de la feuille de calcul *Sales*, ce qui donne la variable rows, qui est un tableau 2D de chaînes, chaque tableau interne représentant une ligne de données ; Ensuite,

nous analysons chacun de ces tableaux internes et les stockons en tant qu'objets *SalesData* - définis par l'interface au bas du script, cela est nécessaire lorsque nous arrivons à sérialiser en JSON (cela garantit que les propriétés des objets *SalesData* (magasin, date, etc.) correspondent aux propriétés dans le JSON résultant) ; enfin, nous sérialisons le tableau résultant d'objets SalesData sous forme de chaîne JSON et renvoyons le résultat!

Voici à quoi ressemble la feuille de calcul des données de ventes principales:

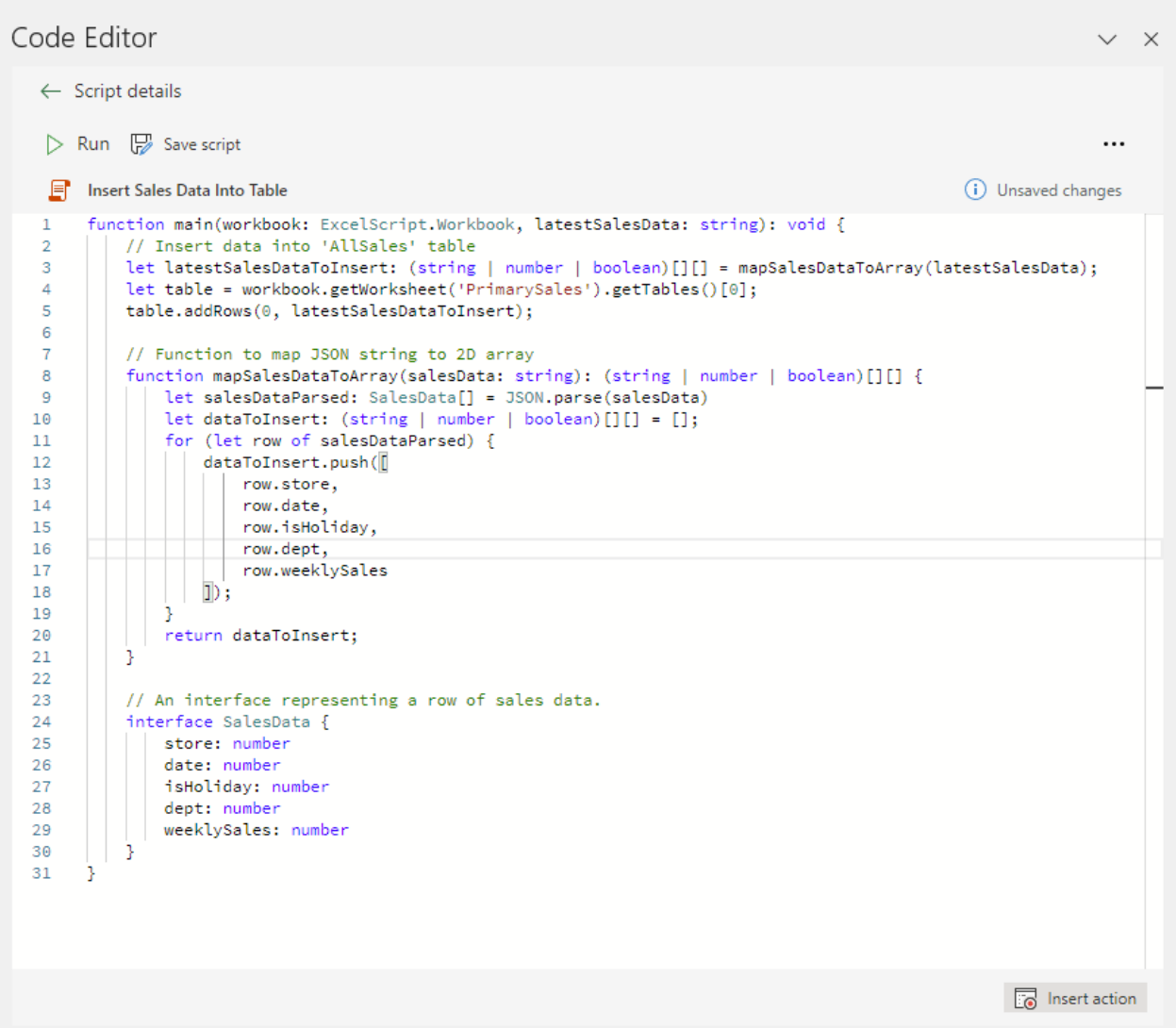


Nous avons maintenant besoin d'un script Office qui prendra les données extraites par le premier script Office et les insérera dans le tableau contenant les données de ventes de l'année entière dans la feuille de calcul principale:

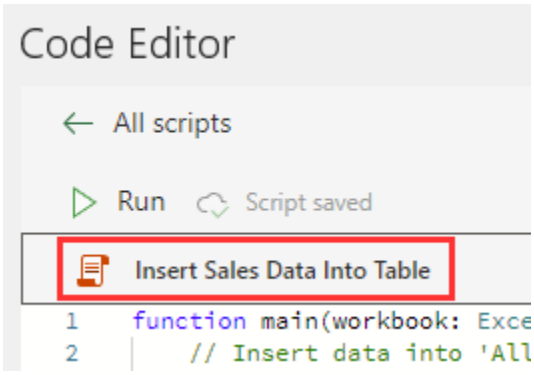


Nous écrivons le code suivant:

Public
Internal



Que nous enregistrons sous le nom:

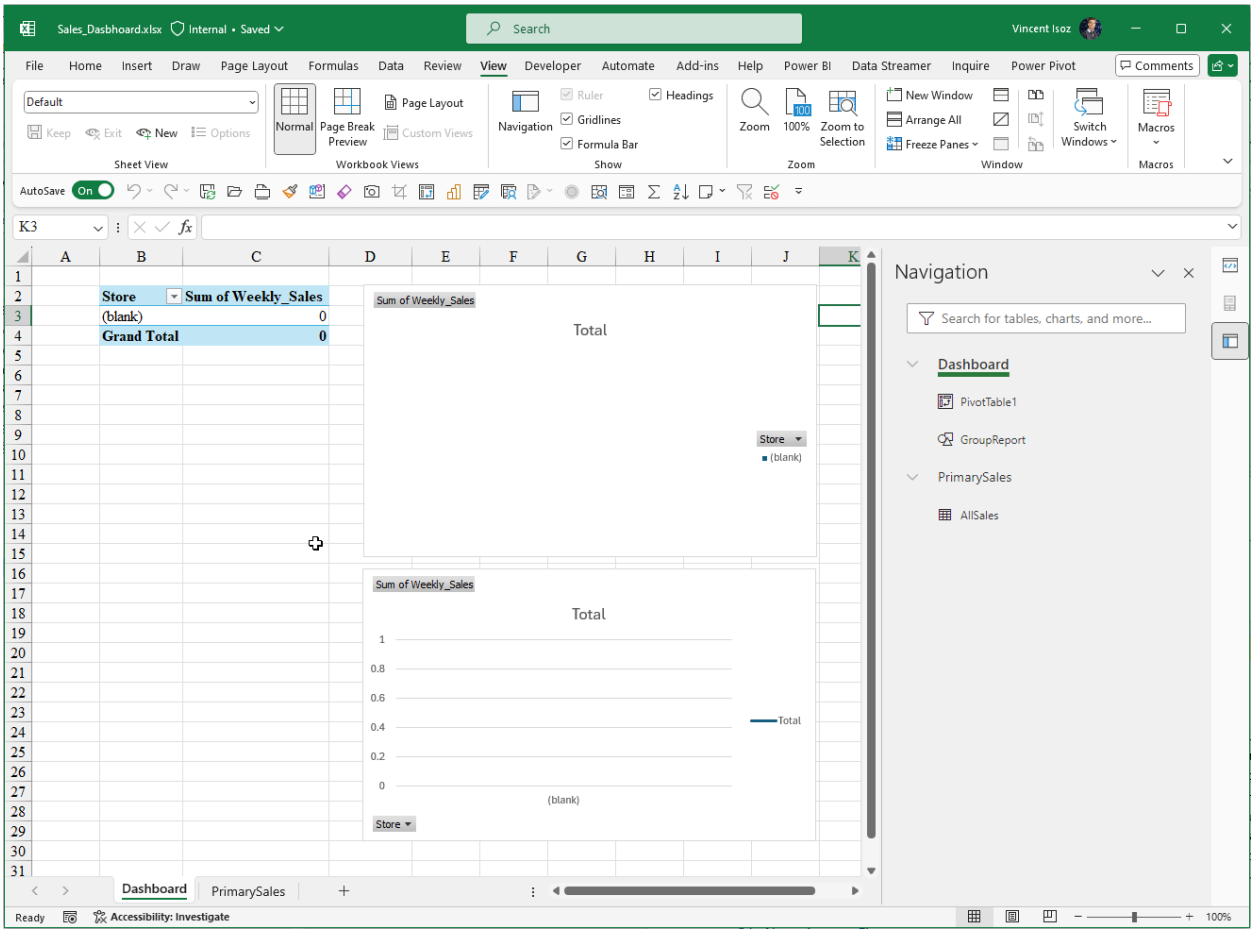


La première chose à remarquer est le paramètre *lastSalesData* pour notre fonction principale de niveau supérieur. La chaîne JSON générée par le premier script Office sera transmise en

argument à ce paramètre (remarque : c'est le flux Power Automate, auquel nous reviendrons plus tard, qui transmet la sortie du premier script en entrée au second).

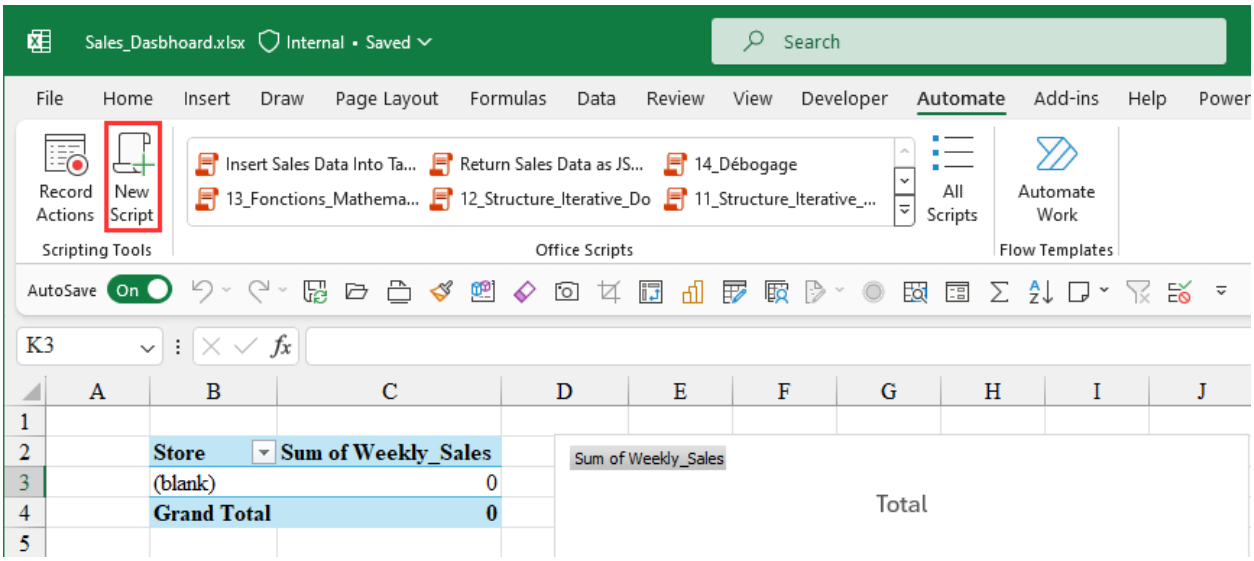
Ce script utilise la fonction *mapSalesDataToArray*, que j'ai définie, pour mapper la chaîne JSON entrante dans un tableau de tableaux (encore une fois, chaque tableau interne représentant une ligne de données), requis par la méthode *addRows*. La fonction utilise d'abord la méthode *JSON.parse* pour analyser la chaîne JSON dans un tableau d'objets *SalesData*, elle parcourt ensuite chacun de ces objets et extrait les données dans un tableau, ce qui donne le tableau de données 2D requis.

Ensuite, nous avons besoin d'un script Office pour extraire des images de graphiques et un tableau récapitulatif, à envoyer au responsable sous forme de rapport. Il y a une autre feuille dans le fichier Excel principal dans laquelle j'ai créé des tableaux croisés dynamiques et des graphiques. Nous avons besoin de ce script Office pour actualiser d'abord les tableaux croisés dynamiques et les graphiques, afin qu'ils contiennent les données insérées, puis capturer les images:



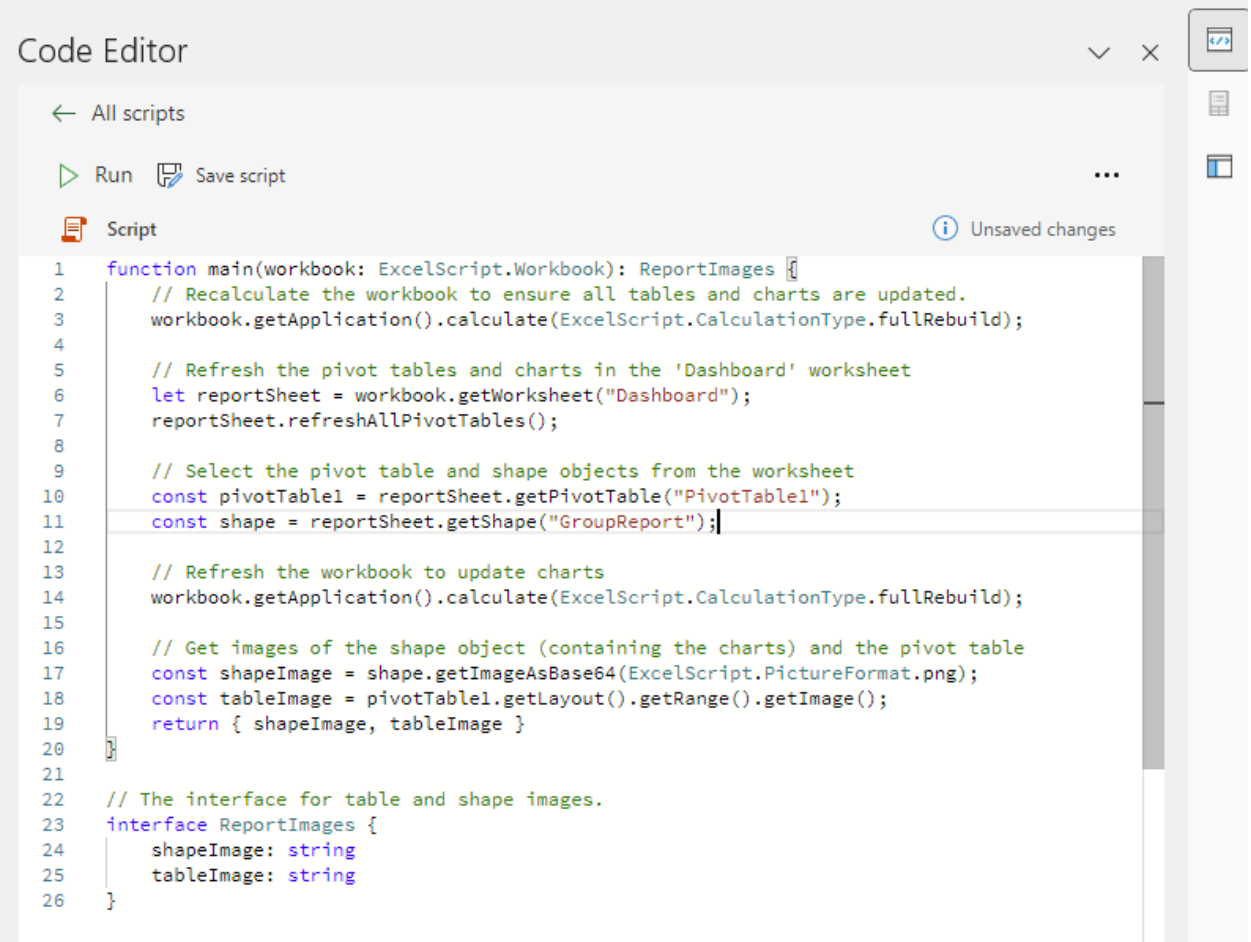
Il existe un tableau croisé dynamique : *PivotTable* ainsi qu'une collection de graphiques qui ont été regroupés, ce qui fait que la collection devient un objet Shape, que j'ai nommé *GroupReport*. Nous souhaitons extraire une image de la collection de graphiques (c'est-à-dire *GroupReport*) et du tableau croisé dynamique en haut à gauche (*PivotTable1*) dans l'image ci-dessus.

Nous créons alors un nouveau script:

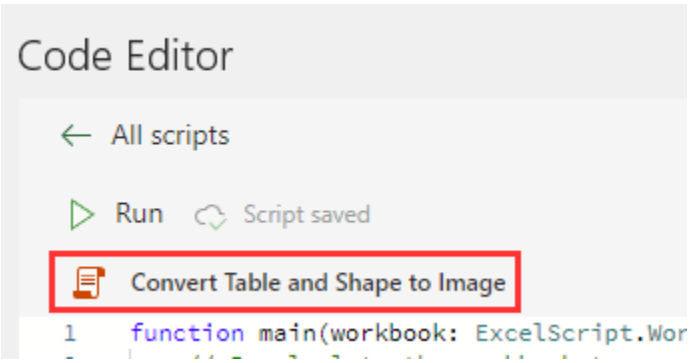


Public
Internal

Et nous écrivons le script suivant:



Et renommons le script:

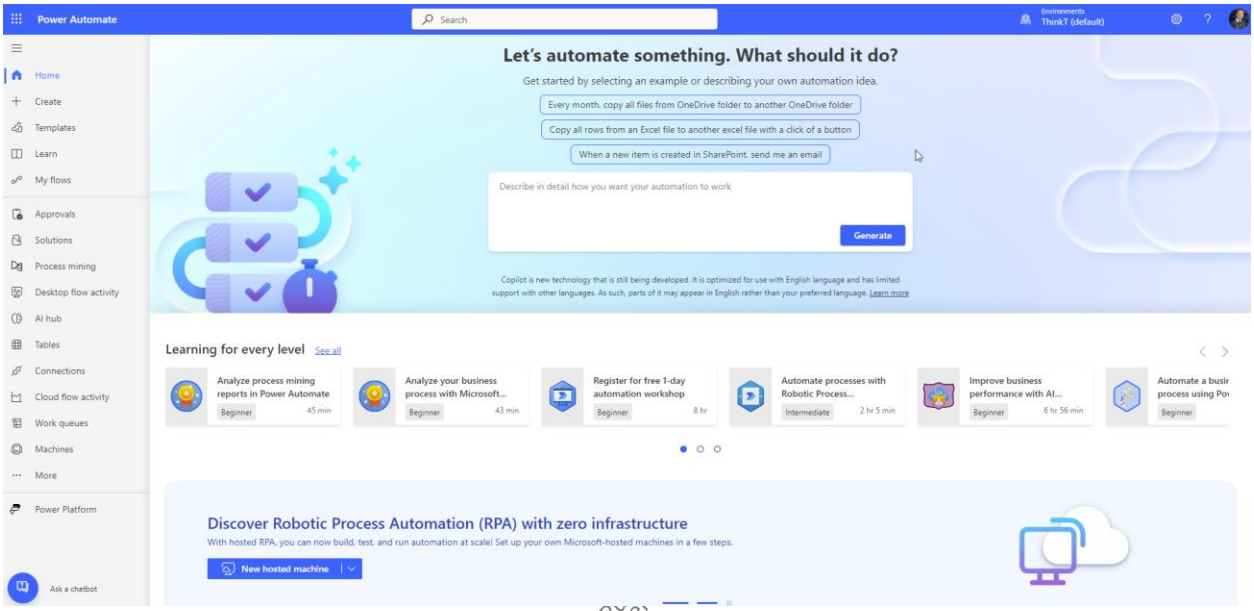


Jetons un coup d'œil au script ci-dessus pour effectuer l'extraction d'image. Nous actualisons d'abord le classeur pour mettre à jour les tableaux et les graphiques, mais cela ne fonctionne pas toujours. J'ai donc ajouté du code pour actualiser explicitement les tableaux croisés dynamiques dans la feuille *Dashboard*. Ensuite, j'actualise à nouveau le classeur afin que le graphique à secteurs soit synchronisé avec le *PivotTable1* (vaut mieux prévenir que guérire…) désormais mis à jour. Enfin, je capture des images de *PivotTable1* et de la forme *GroupReport* (contenant les

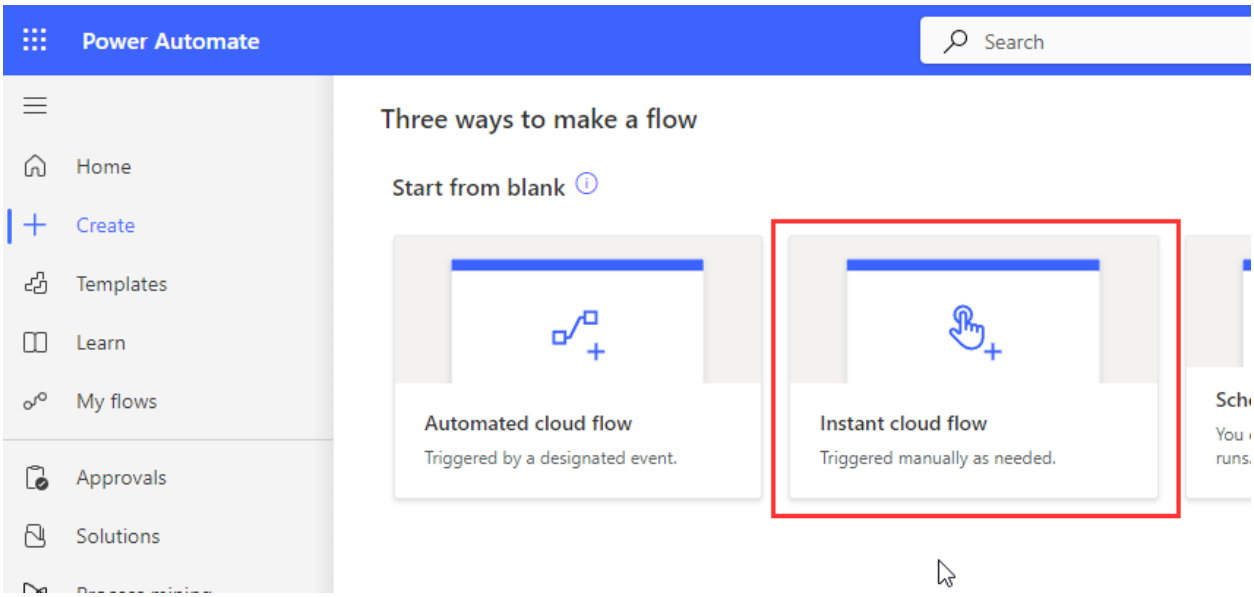
graphiques), qui sont codées sous forme de chaînes, et je les renvoie en tant que propriétés sur un objet *ReportImages* pour Power Automate.

Fermez tous les fichiers Microsoft Excel à partir de maintenant!

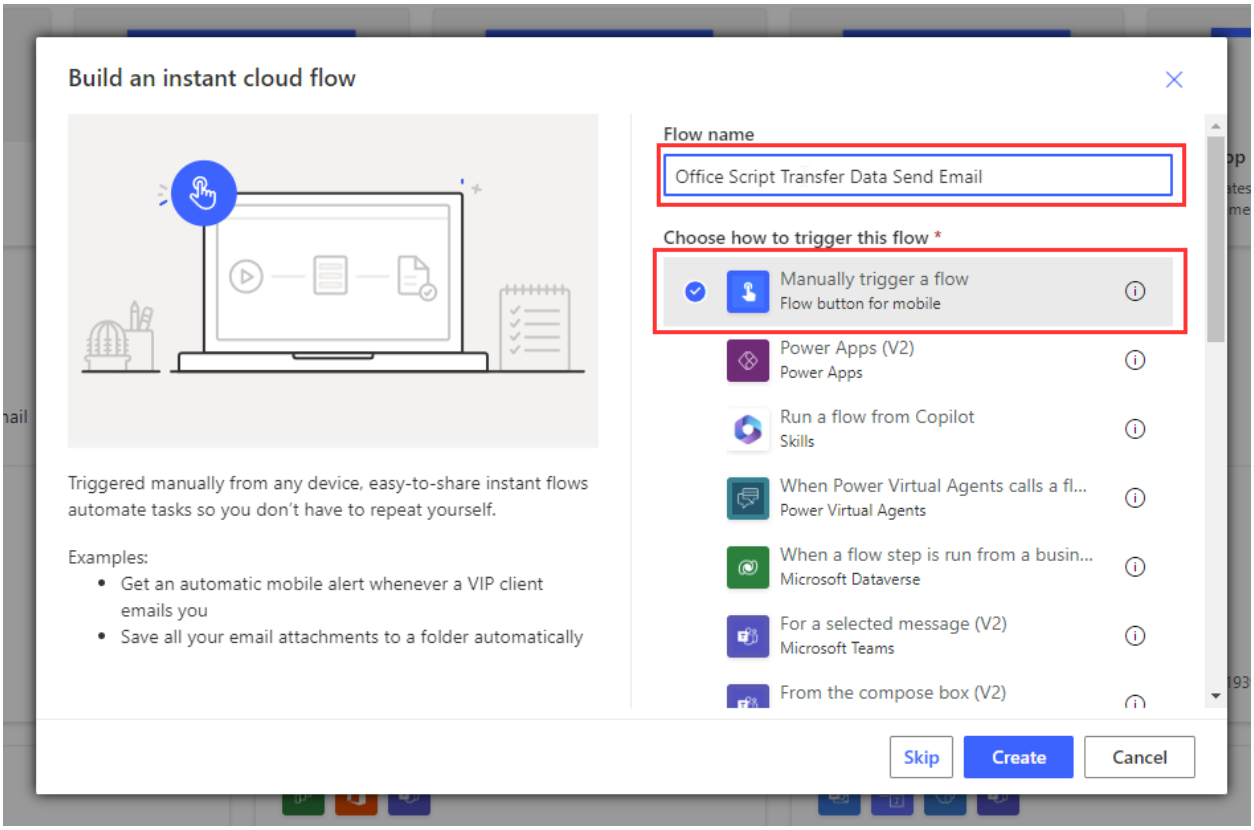
Maintenant, nous allons dans Power Automate:

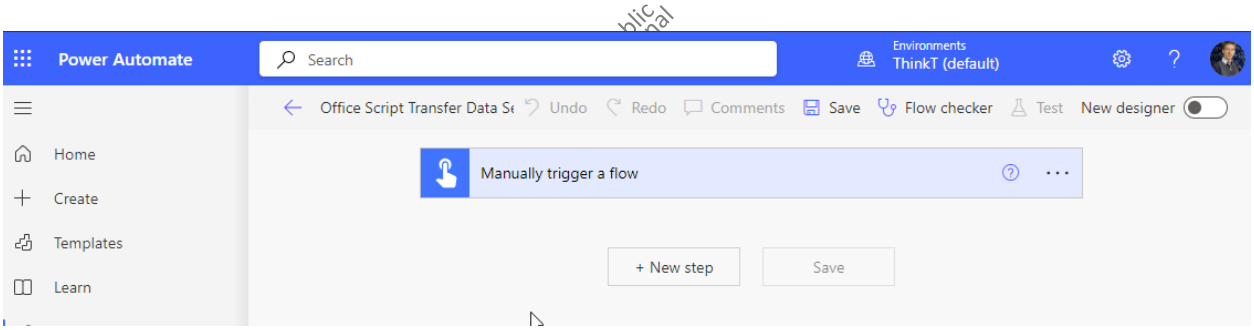


Nous cliquons sur **Create** et prenons **Instant cloud flow**:

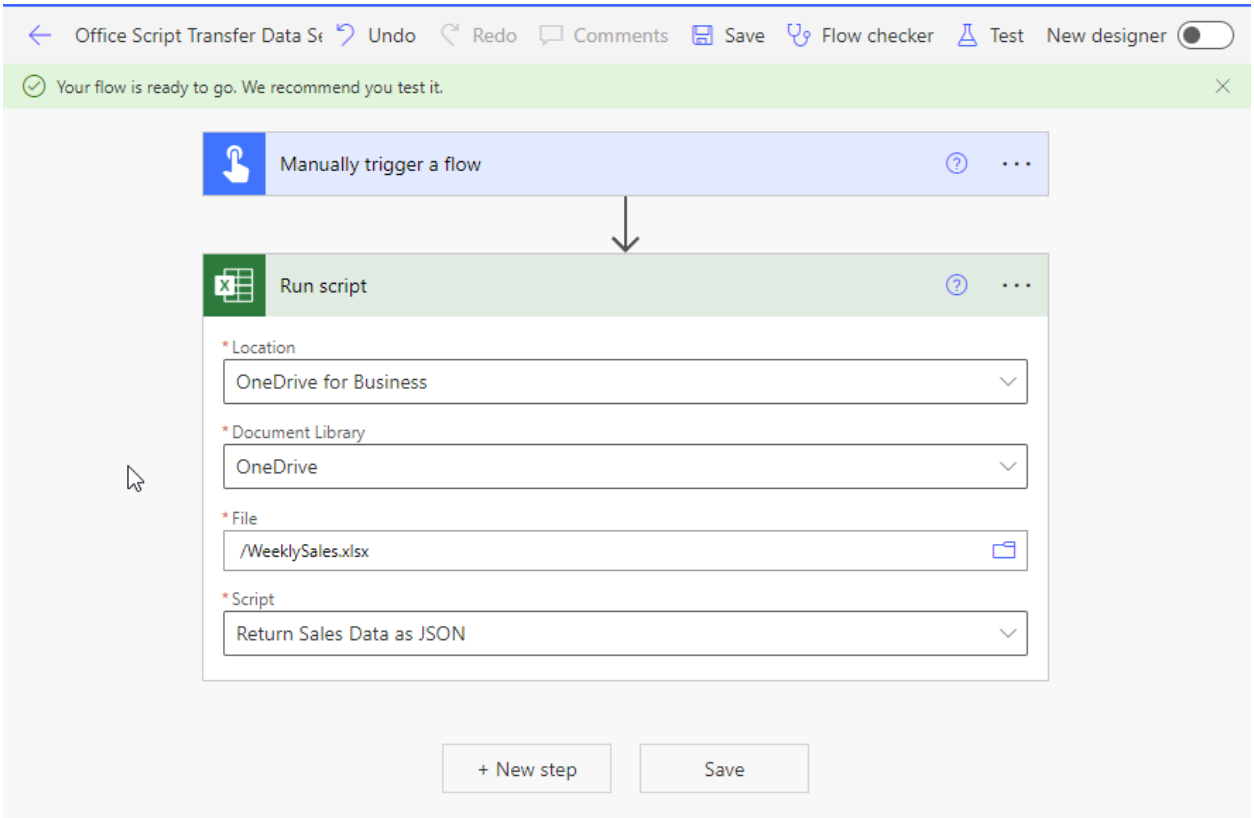


Et nommons le de la manière suivante:



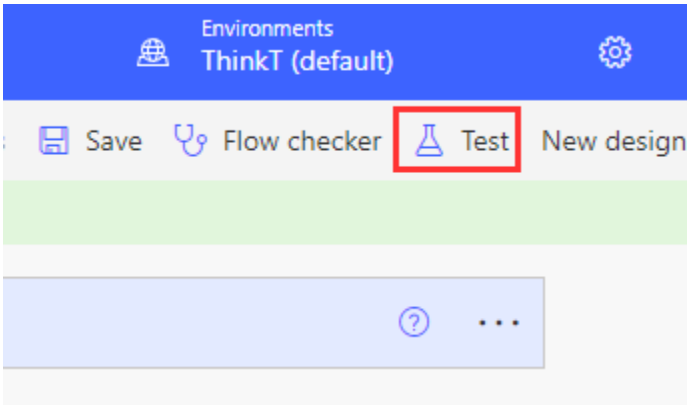


Nous créons l'action suivante:

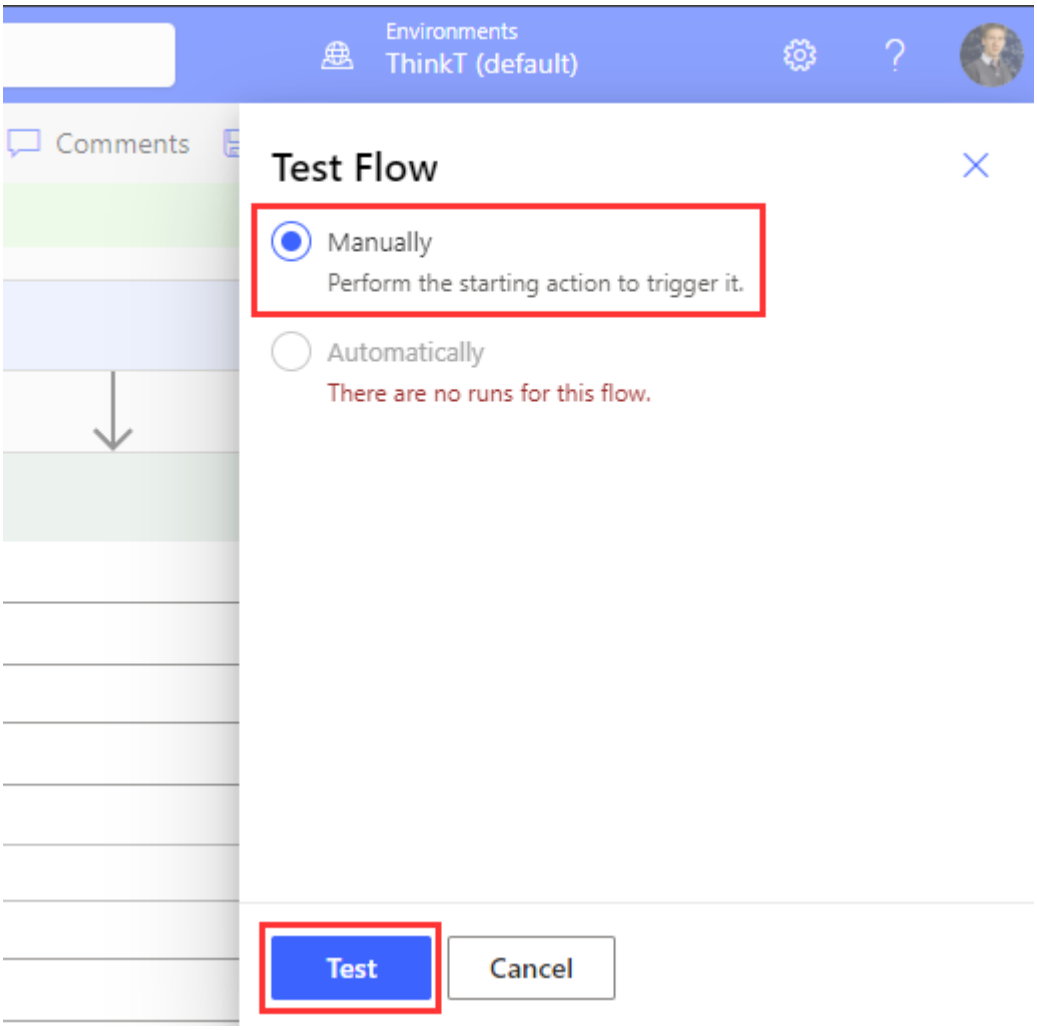


Nous cliquons sur **Test**:

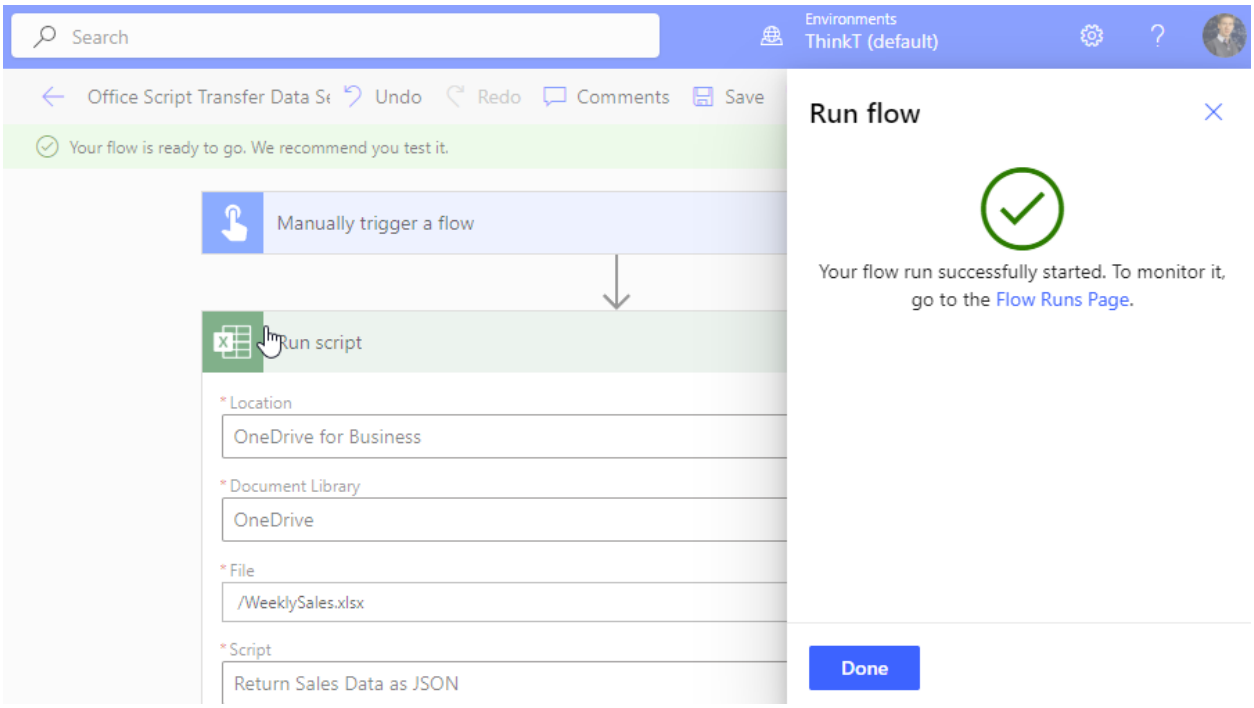
Public Internal



Et nous prenons *Manually* et nous cliquons sur **Test**:

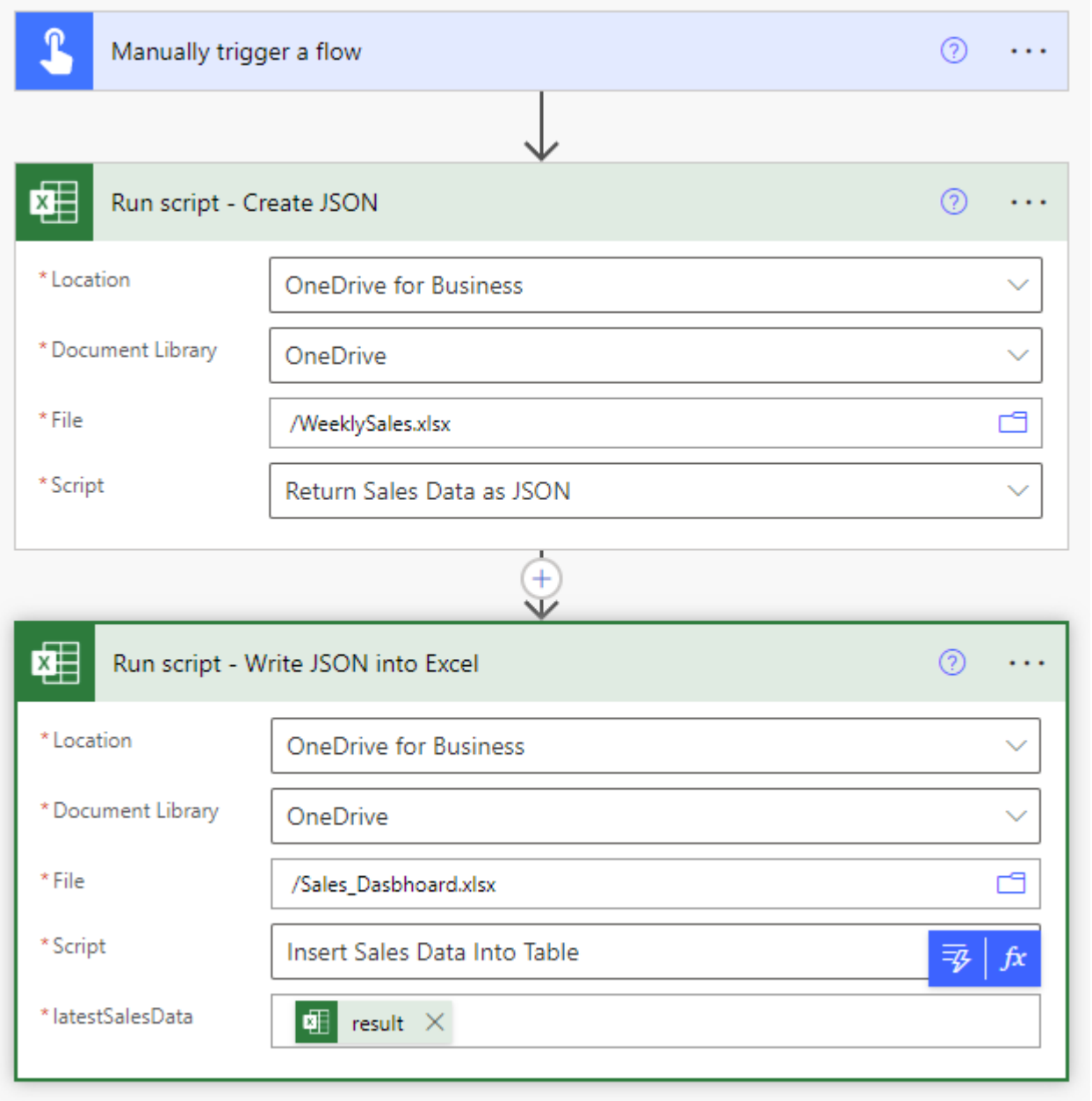


Nous aurons alors si nous n'avons pas fait d'erreurs:

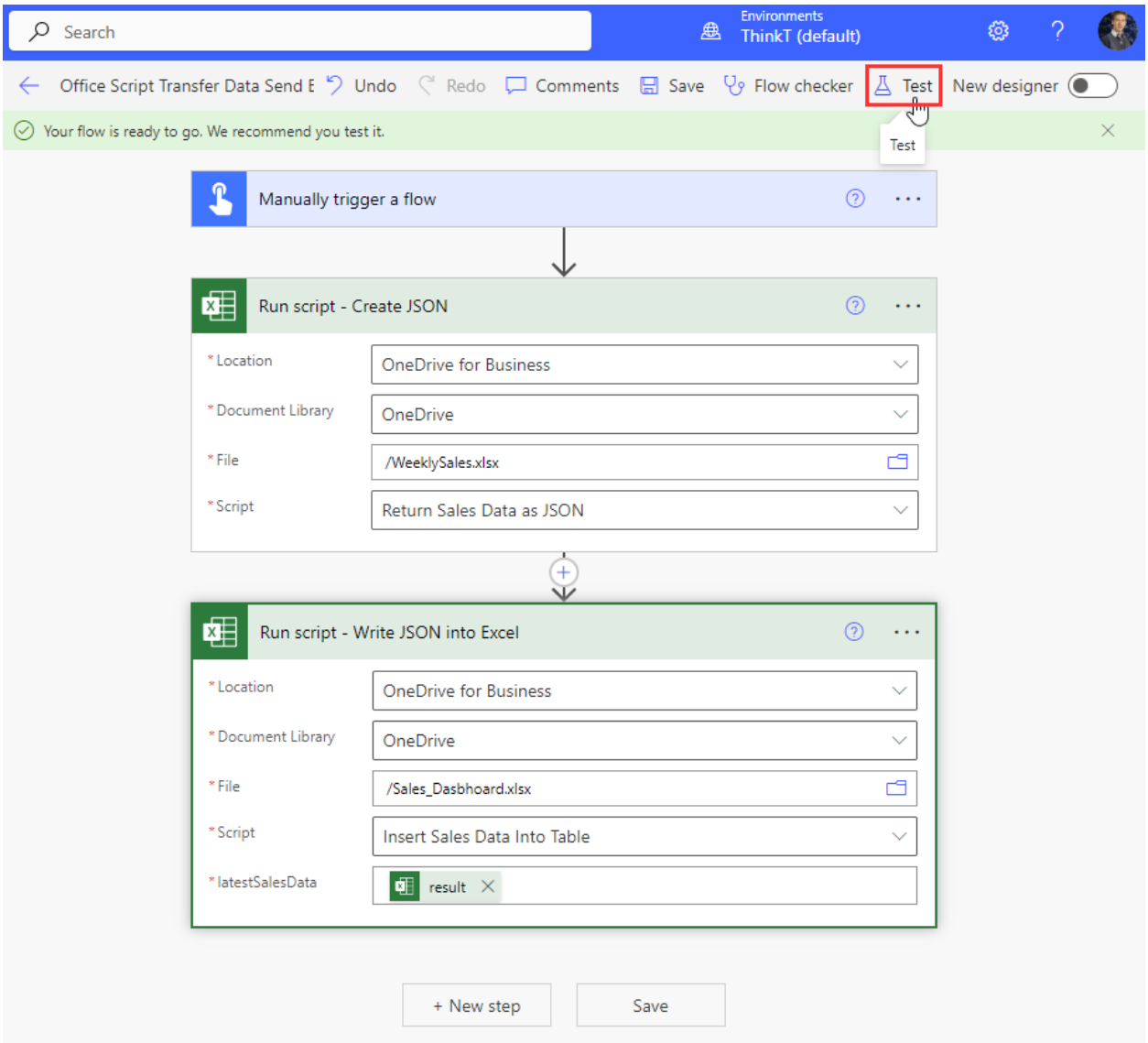


Donc jusque là cela semble fonctionner mais comme cela n'écrit rien où que ce soit, nous ne pouvons rien ouvrir pour vérifier que cela a marché. Donc ajoutons la prochaine étape (et profitons-en pour renommer les étapes avec un nom plus sympa):

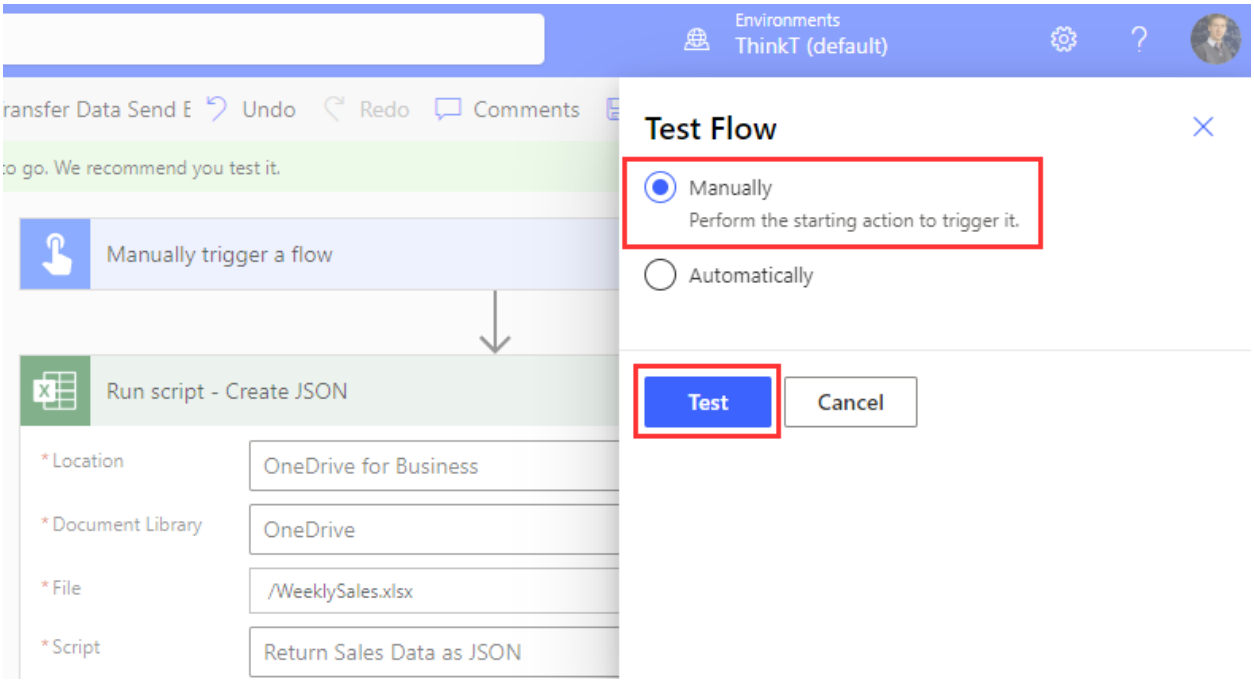
Public Interne



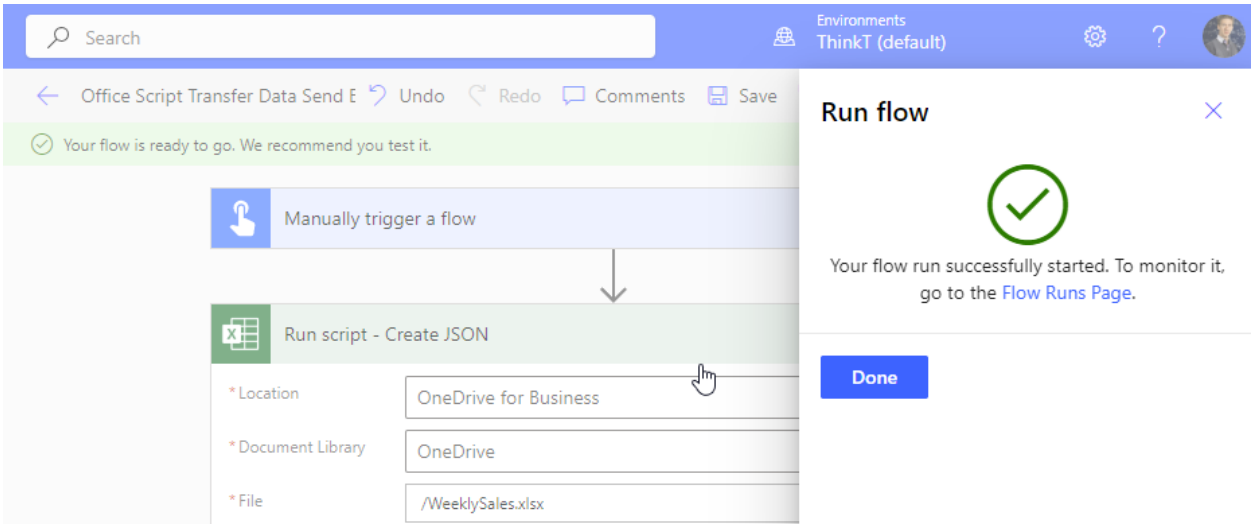
Nous testons à nouveau:



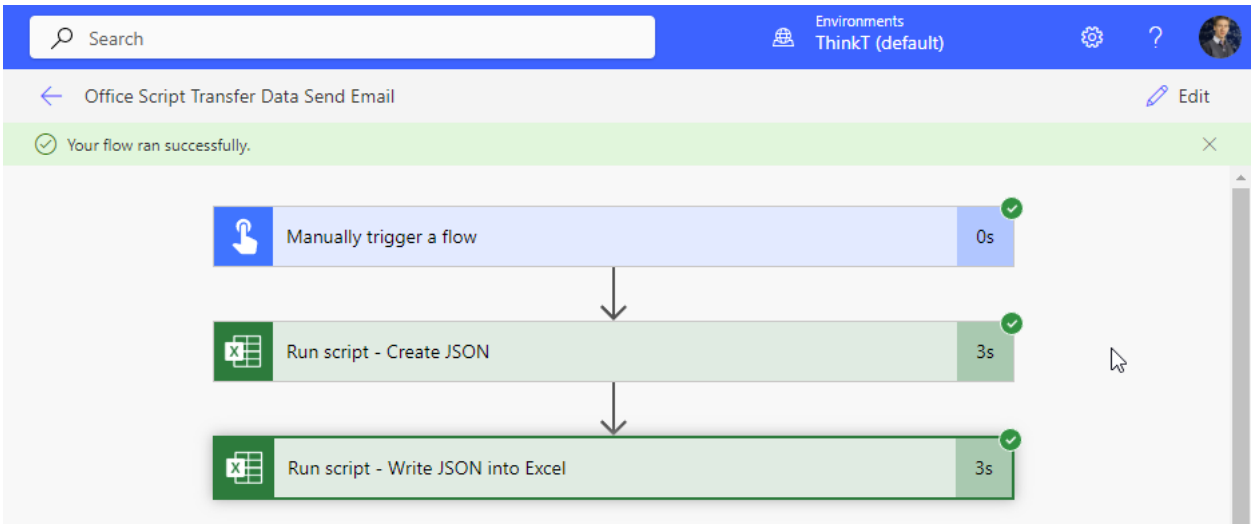
Nous nous prenons *Manually* et cliquons sur *Test*:



Si nous avons bien fait les choses, nous aurons:



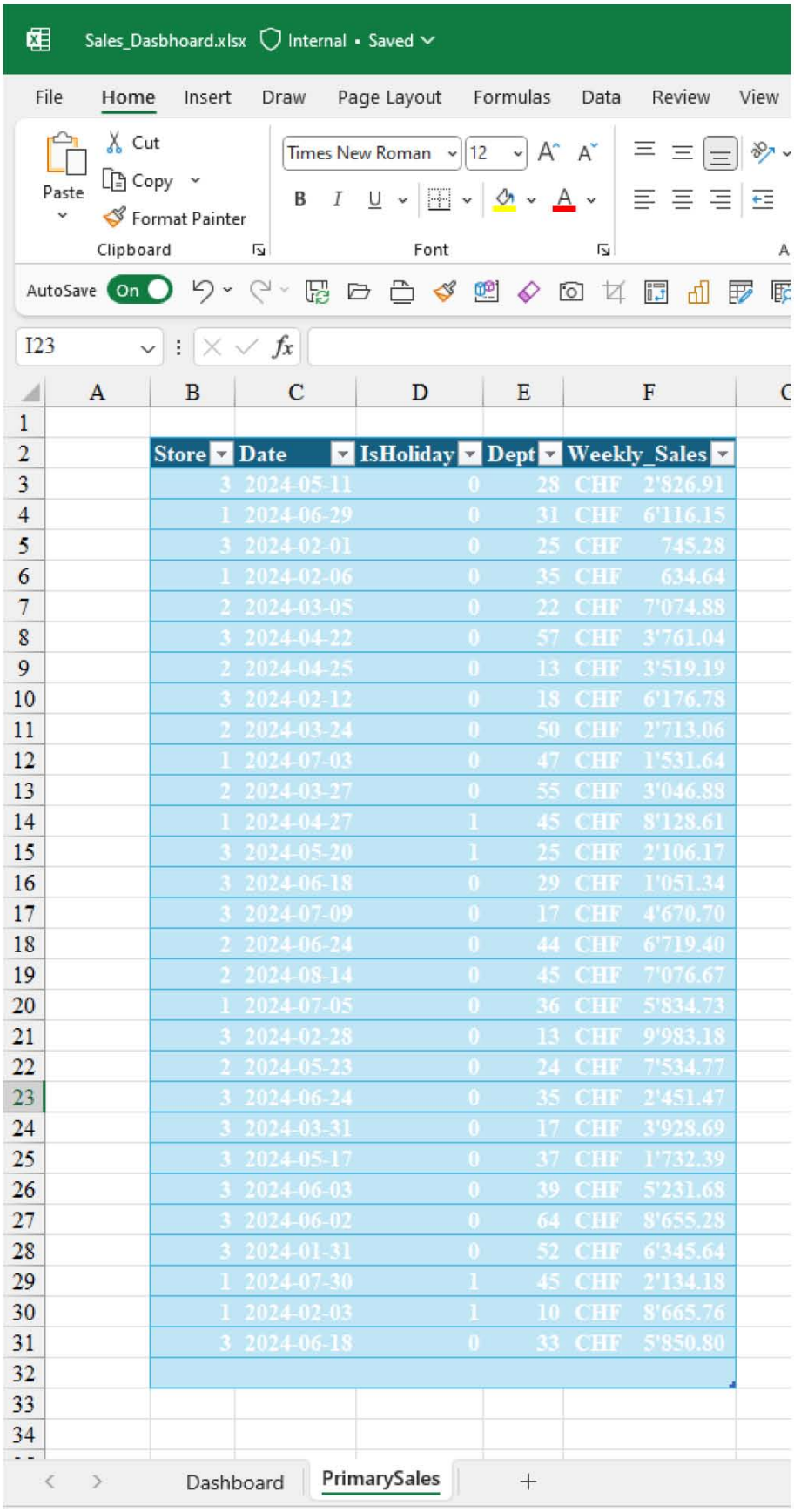
Soit:



Si nous ouvrons maintenant le fichier:

Nous aurons:

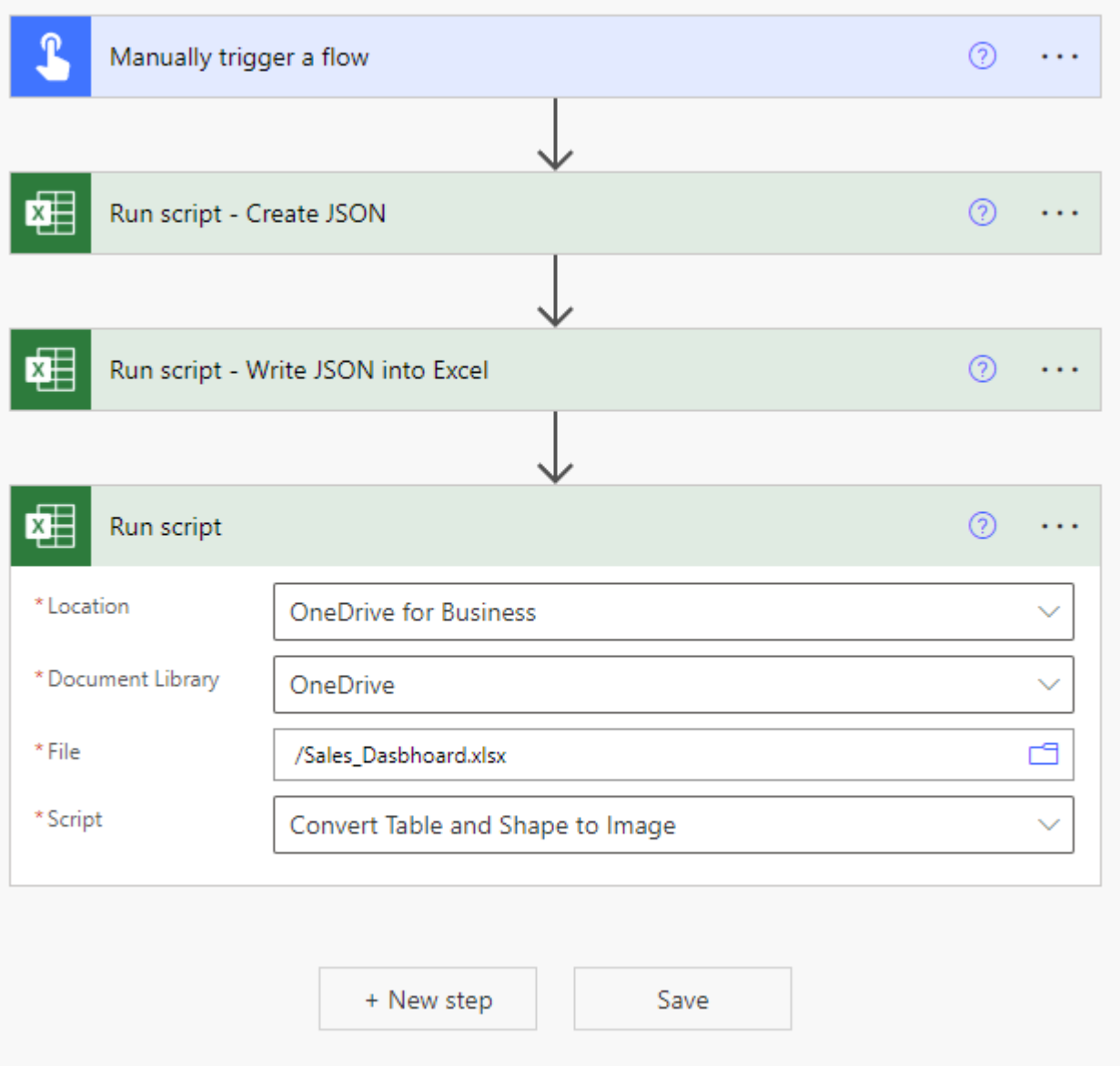
Public
Internal



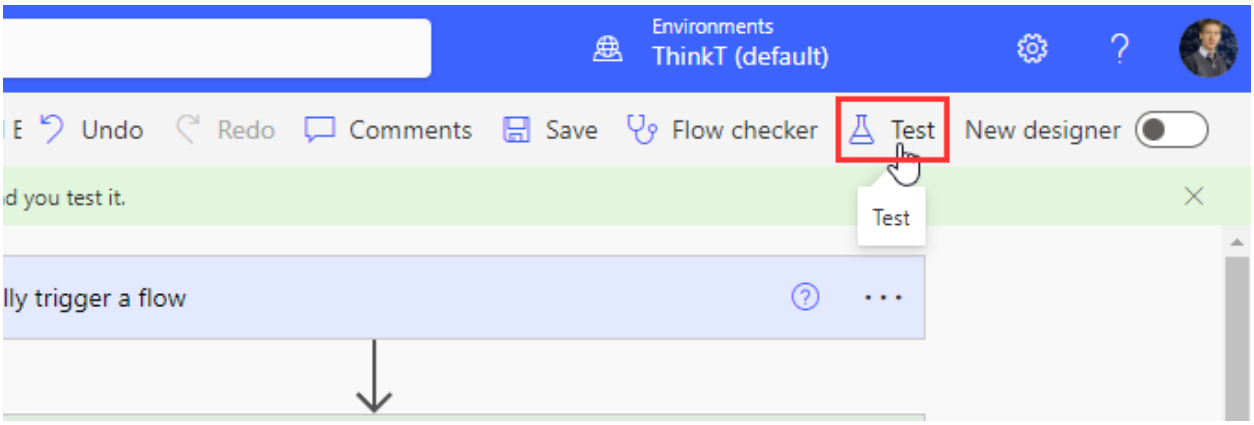
Store	Date	IsHoliday	Dept	Weekly_Sales
3	2024-05-11	0	28	CHF 2'826.91
1	2024-06-29	0	31	CHF 6'116.15
3	2024-02-01	0	25	CHF 745.28
1	2024-02-06	0	35	CHF 634.64
2	2024-03-05	0	22	CHF 7'074.88
3	2024-04-22	0	57	CHF 3'761.04
2	2024-04-25	0	13	CHF 3'519.19
3	2024-02-12	0	18	CHF 6'176.78
2	2024-03-24	0	50	CHF 2'713.06
1	2024-07-03	0	47	CHF 1'531.64
2	2024-03-27	0	55	CHF 3'046.88
1	2024-04-27	1	45	CHF 8'128.61
3	2024-05-20	1	25	CHF 2'106.17
3	2024-06-18	0	29	CHF 1'051.34
3	2024-07-09	0	17	CHF 4'670.70
2	2024-06-24	0	44	CHF 6'719.40
2	2024-08-14	0	45	CHF 7'076.67
1	2024-07-05	0	36	CHF 5'834.73
3	2024-02-28	0	13	CHF 9'983.18
2	2024-05-23	0	24	CHF 7'534.77
3	2024-06-24	0	35	CHF 2'451.47
3	2024-03-31	0	17	CHF 3'928.69
3	2024-05-17	0	37	CHF 1'732.39
3	2024-06-03	0	39	CHF 5'231.68
3	2024-06-02	0	64	CHF 8'655.28
3	2024-01-31	0	52	CHF 6'345.64
1	2024-07-30	1	45	CHF 2'134.18
1	2024-02-03	1	10	CHF 8'665.76
3	2024-06-18	0	33	CHF 5'850.80

Donc les données ont bien été copiées!

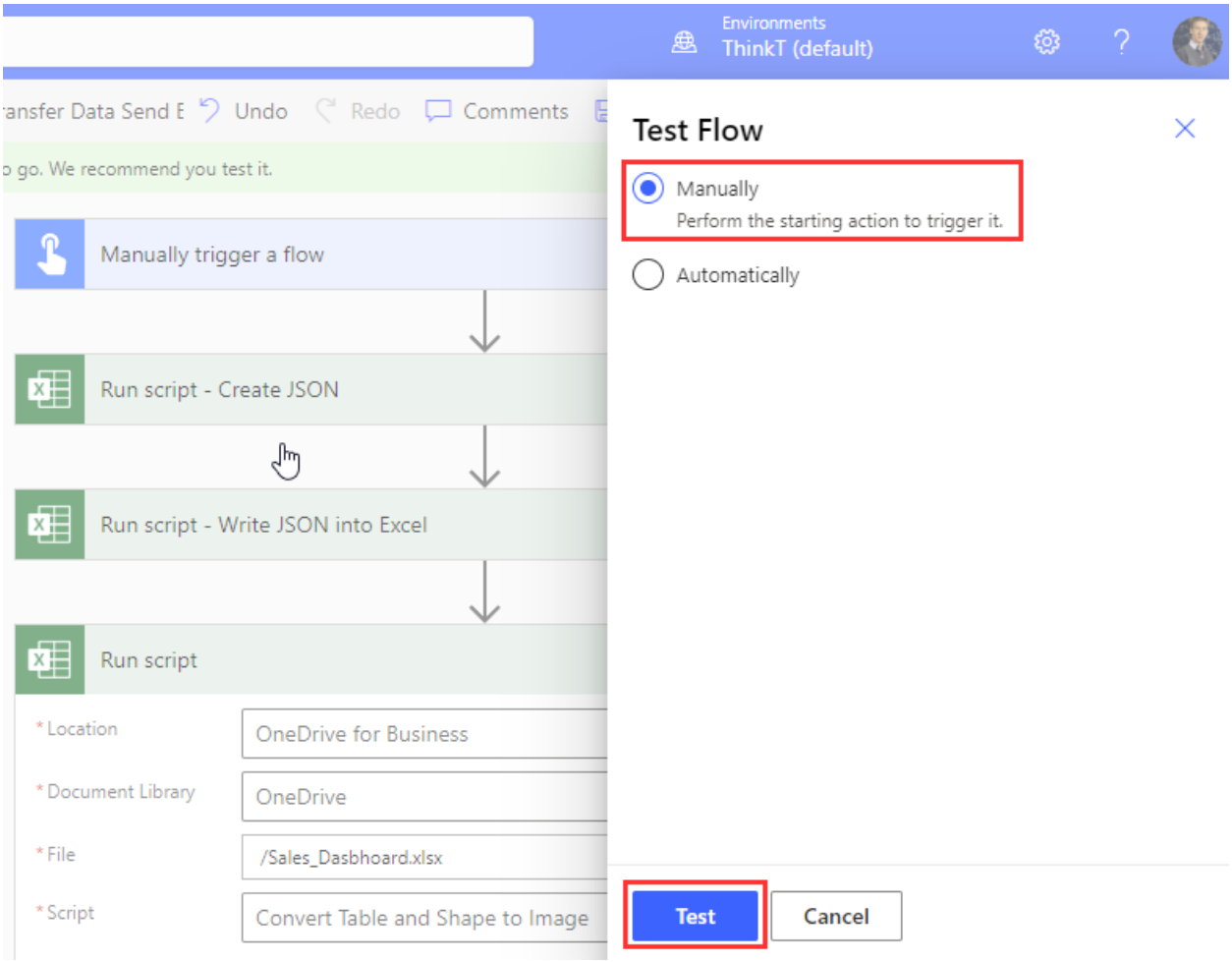
Maintenant ajoutons la dernière exécution de script (celle qui va créer les images):



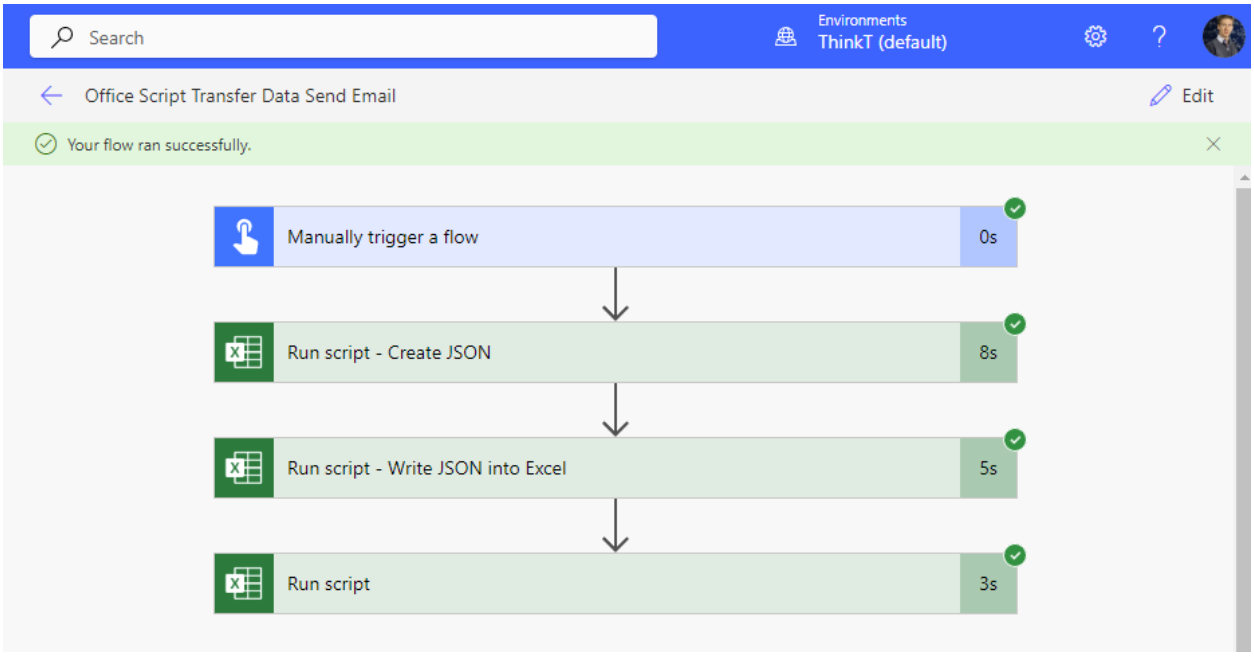
Encore une fois, nous testons:



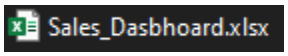
Et nous validons en cliquant sur **Test**:



Si vous n'avez pas fait d'erreurs, vous aurez alors:

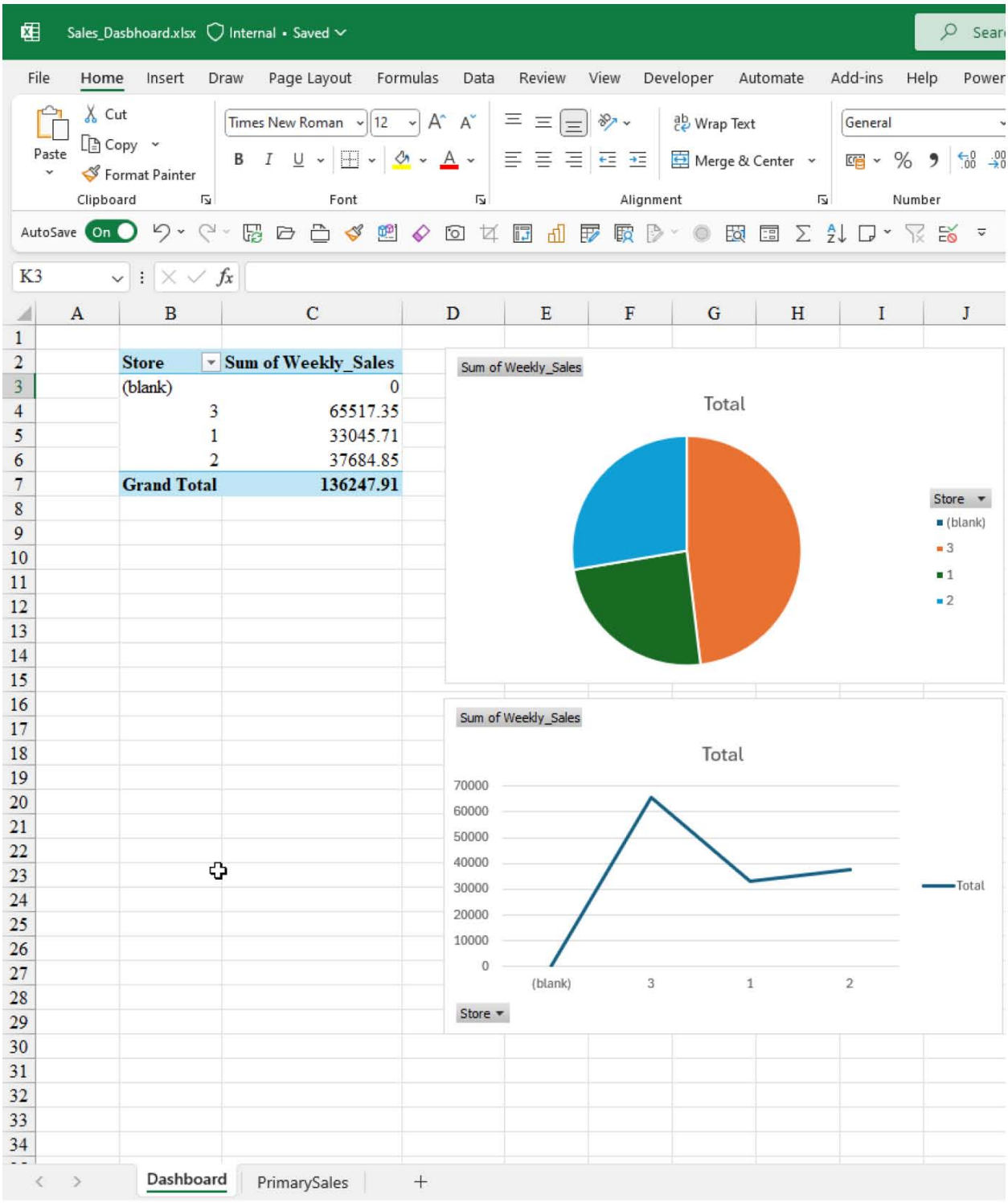


Et en ouvrant le fichier:

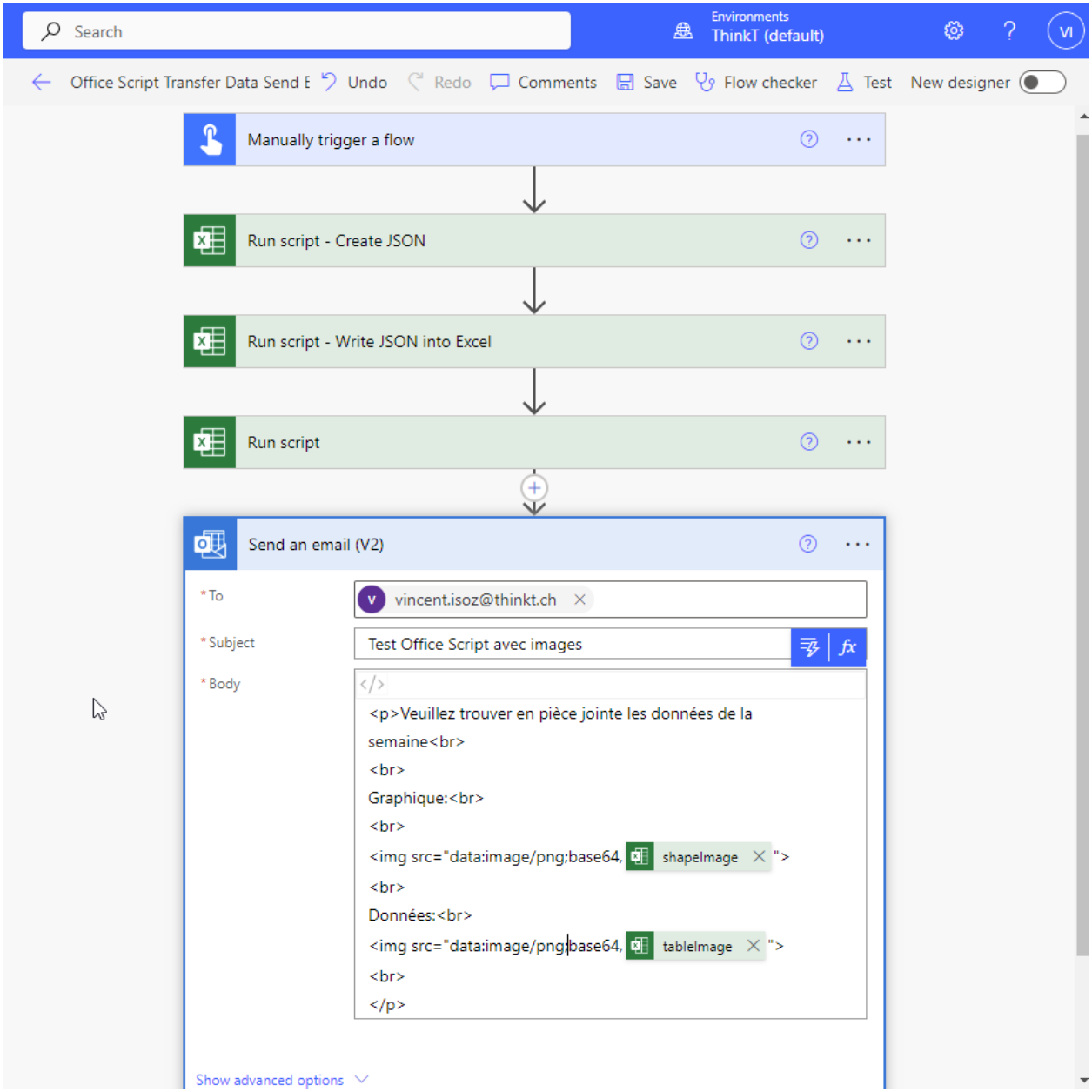


Vous aurez alors:

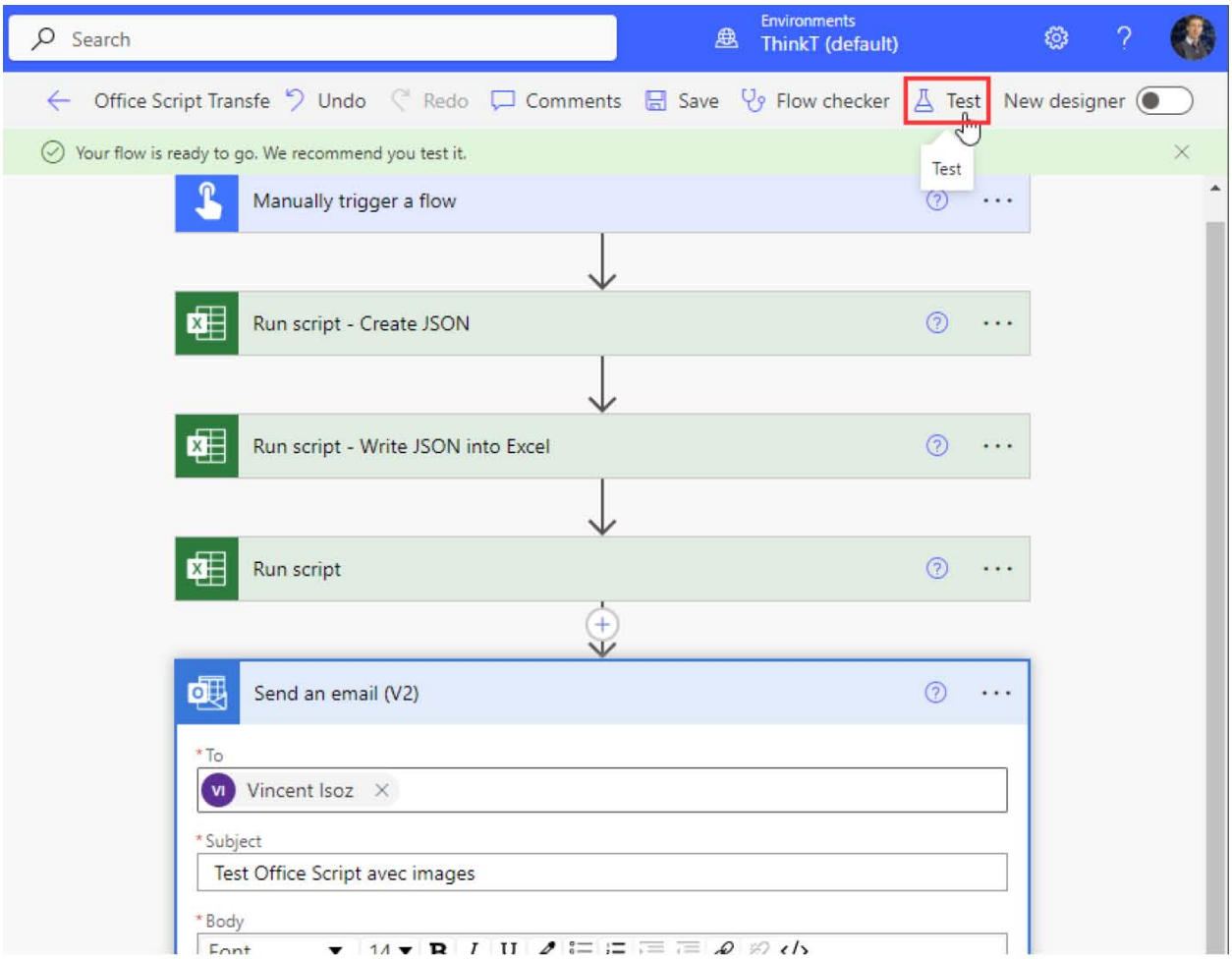
Public
Internal



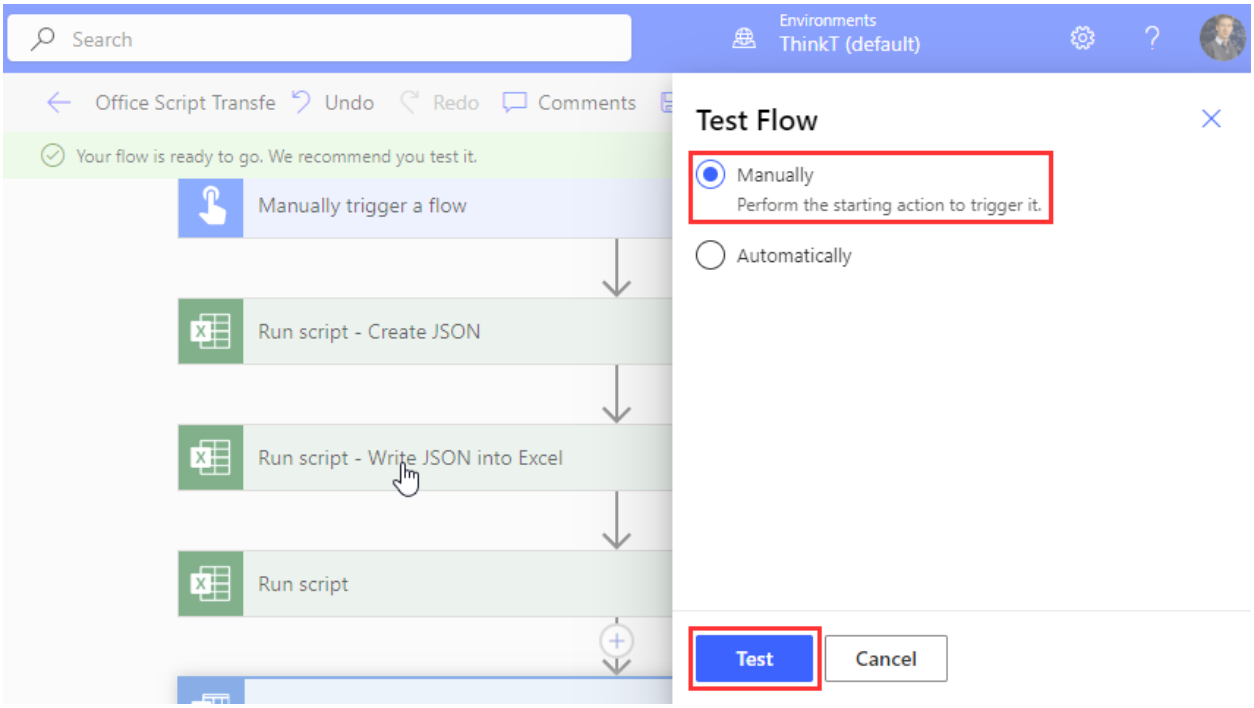
Maintenant créons la dernière action du Flow la plus challenging:



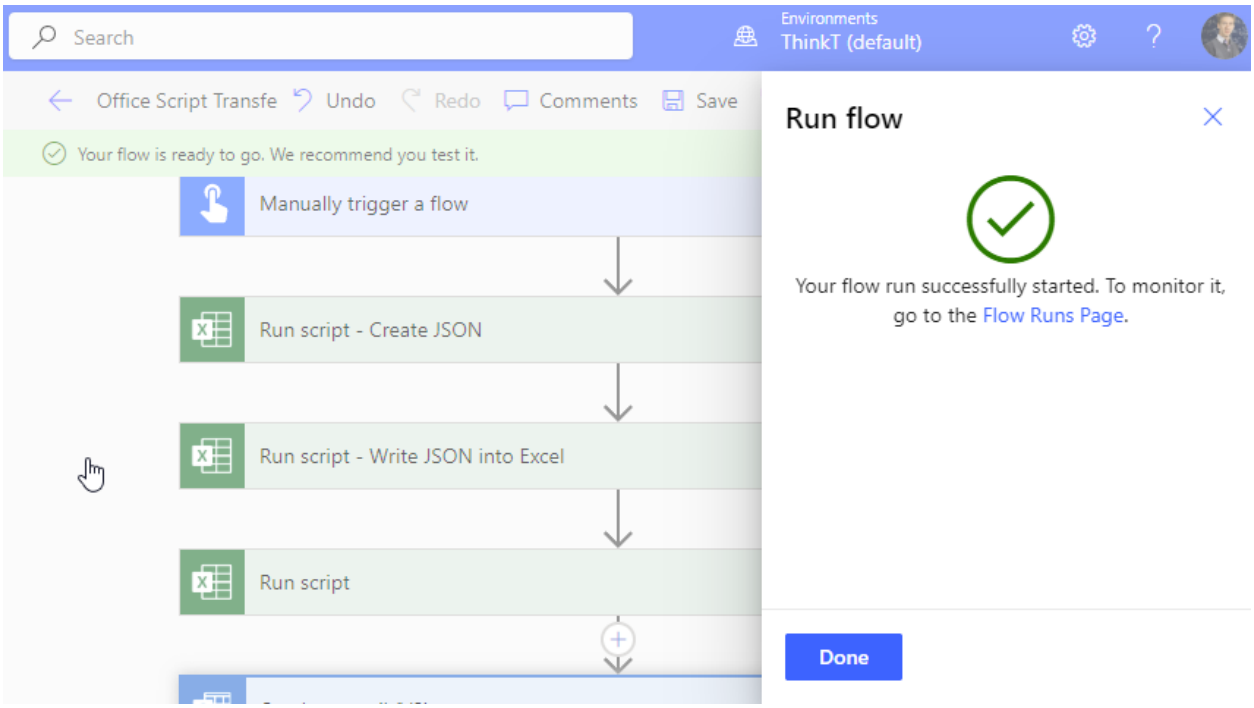
Et pour la dernière fois nous testons:


Search
Environments
ThinkT (default)
Office Script Transfe
Undo
Redo
Comments
Save
Flow checker
Test
New designer
Your flow is ready to go. We recommend you test it.
Test
Manually trigger a flow
Run script - Create JSON
Run script - Write JSON into Excel
Run script
Send an email (V2)
*To
VI
Vincent Isoz
*Subject
Test Office Script avec images
*Body

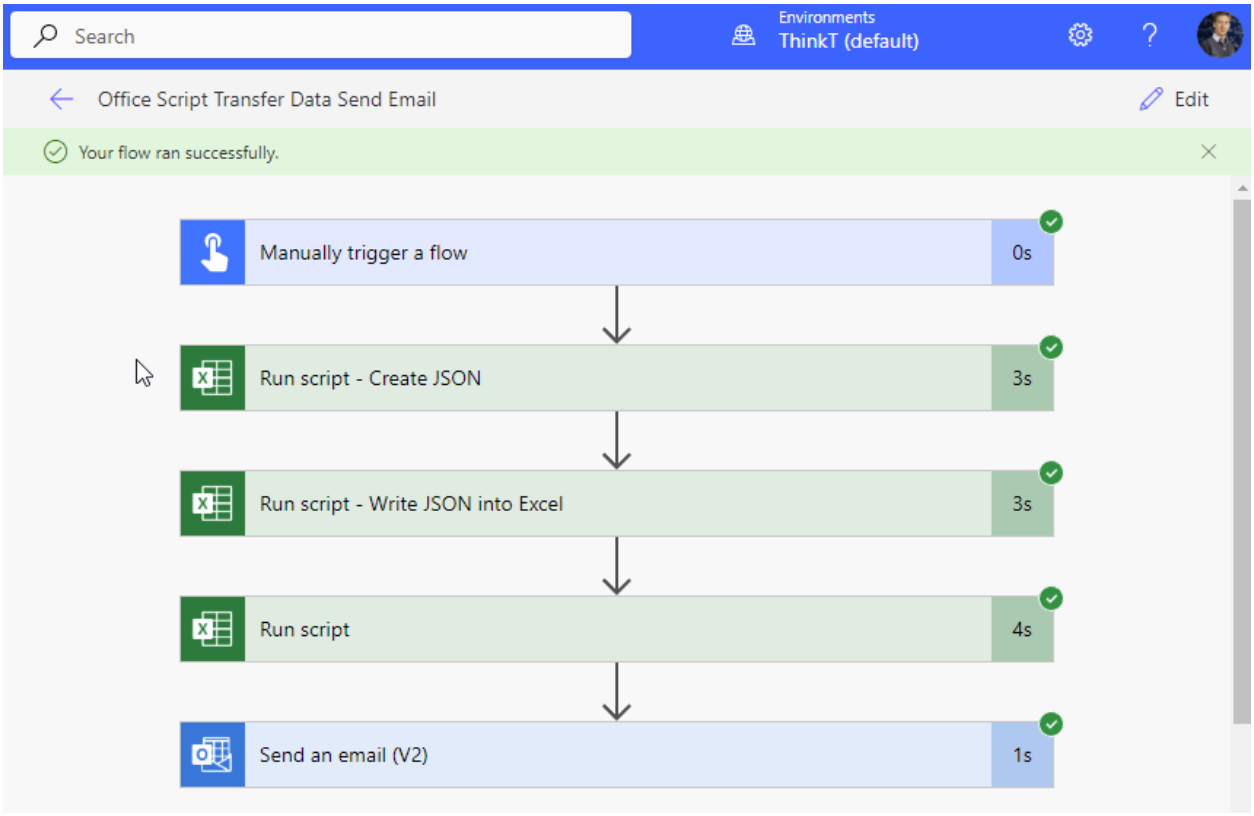
Soit:



Ce qui donne après avoir cliqué sur **Test**:

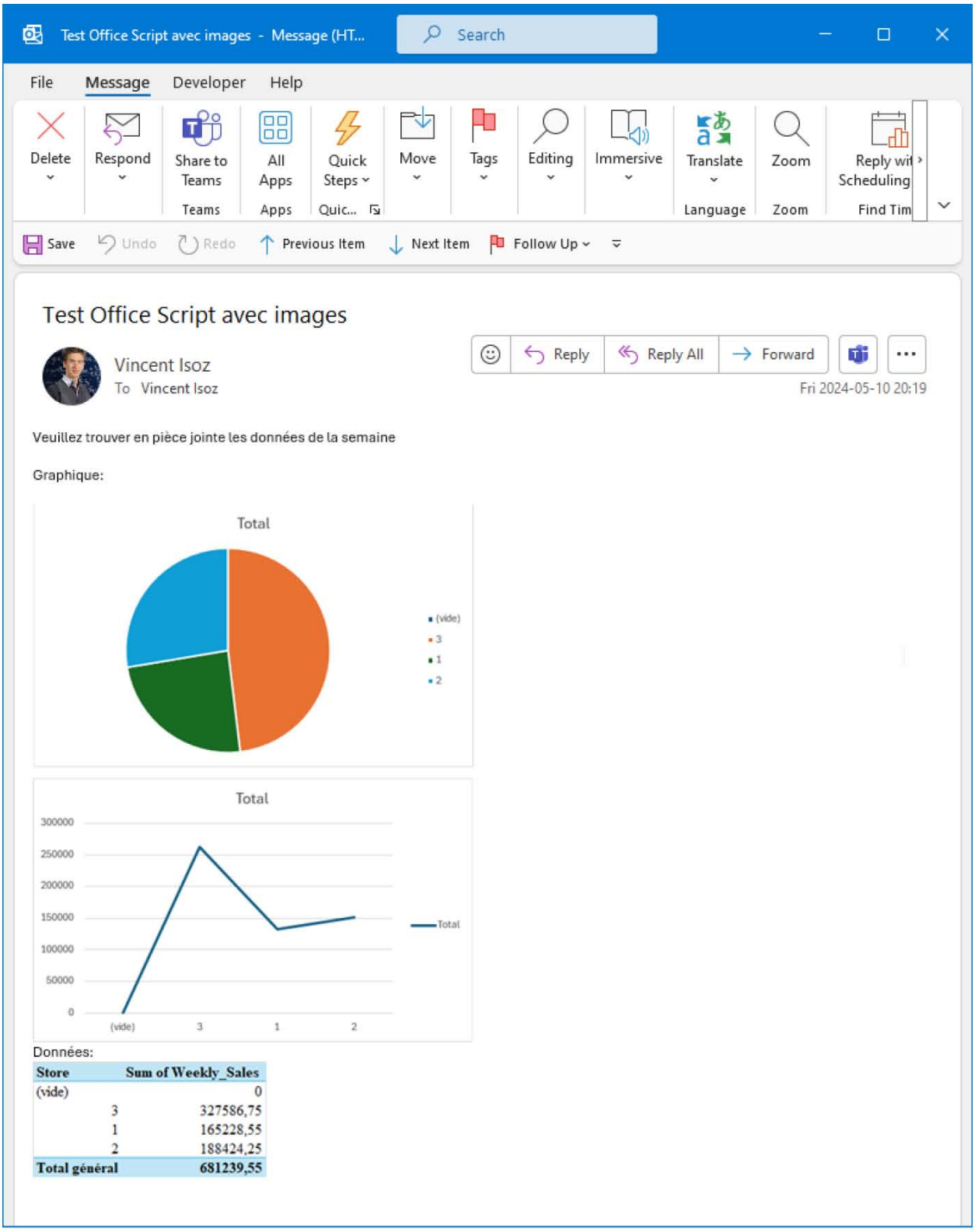


Soit si tout se passe bien:



Et donc vous devriez avoir un e-mail dans votre boîte de réception avec:

Public Internal

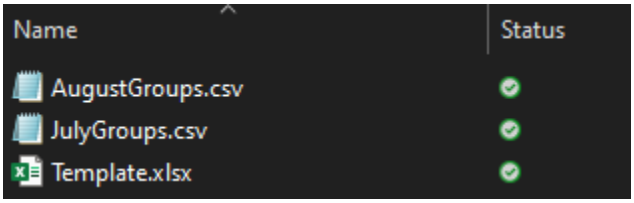


Store	Sum of Weekly_Sales
(vide)	0
3	327586,75
1	165228,55
2	188424,25
Total général	**681239,55**

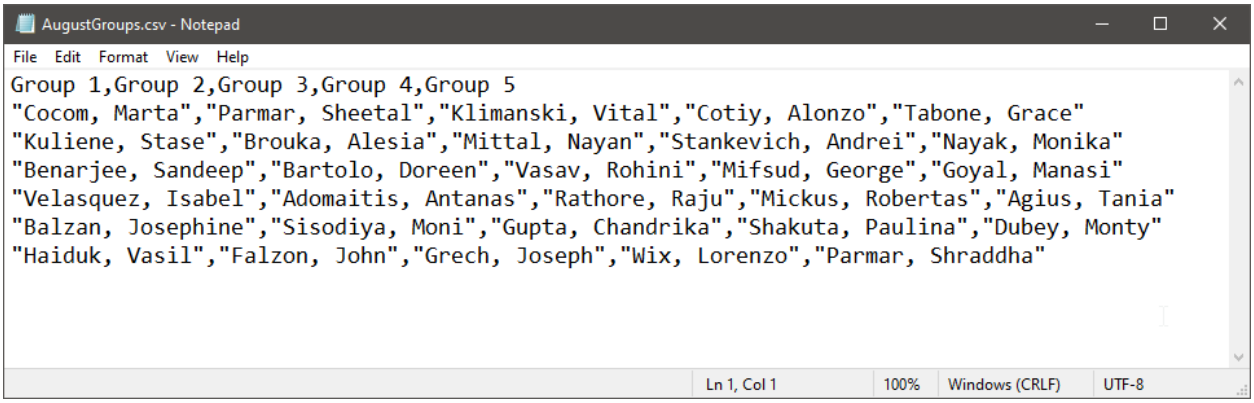
14.3 Conversion de fichiers *.csv en fichiers Excel (copie de données)

Considérons le dossier suivant disponible dans le dossier d'exercice de la formation préalablement déplacé dans votre OneDrive:

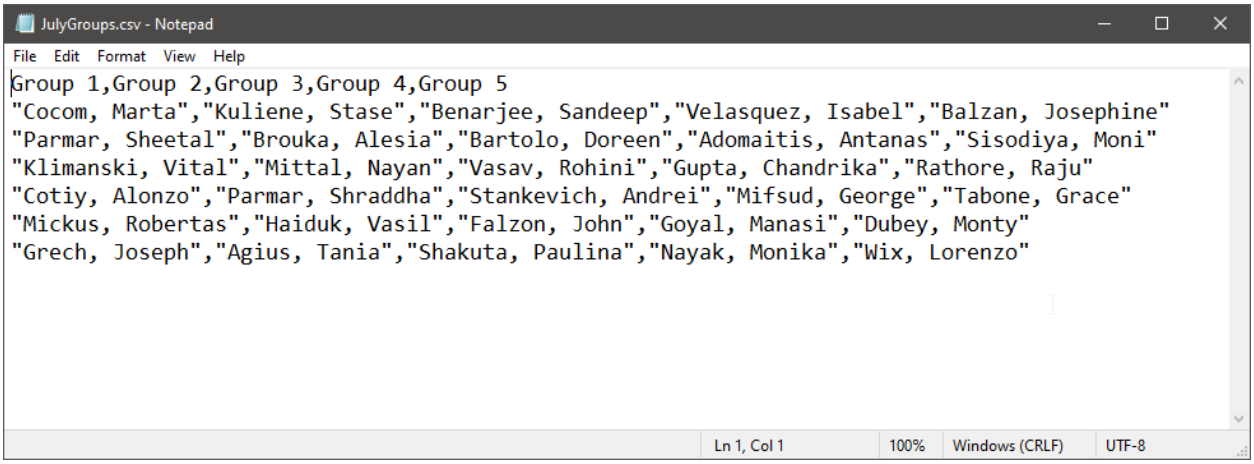
Avec dedans:



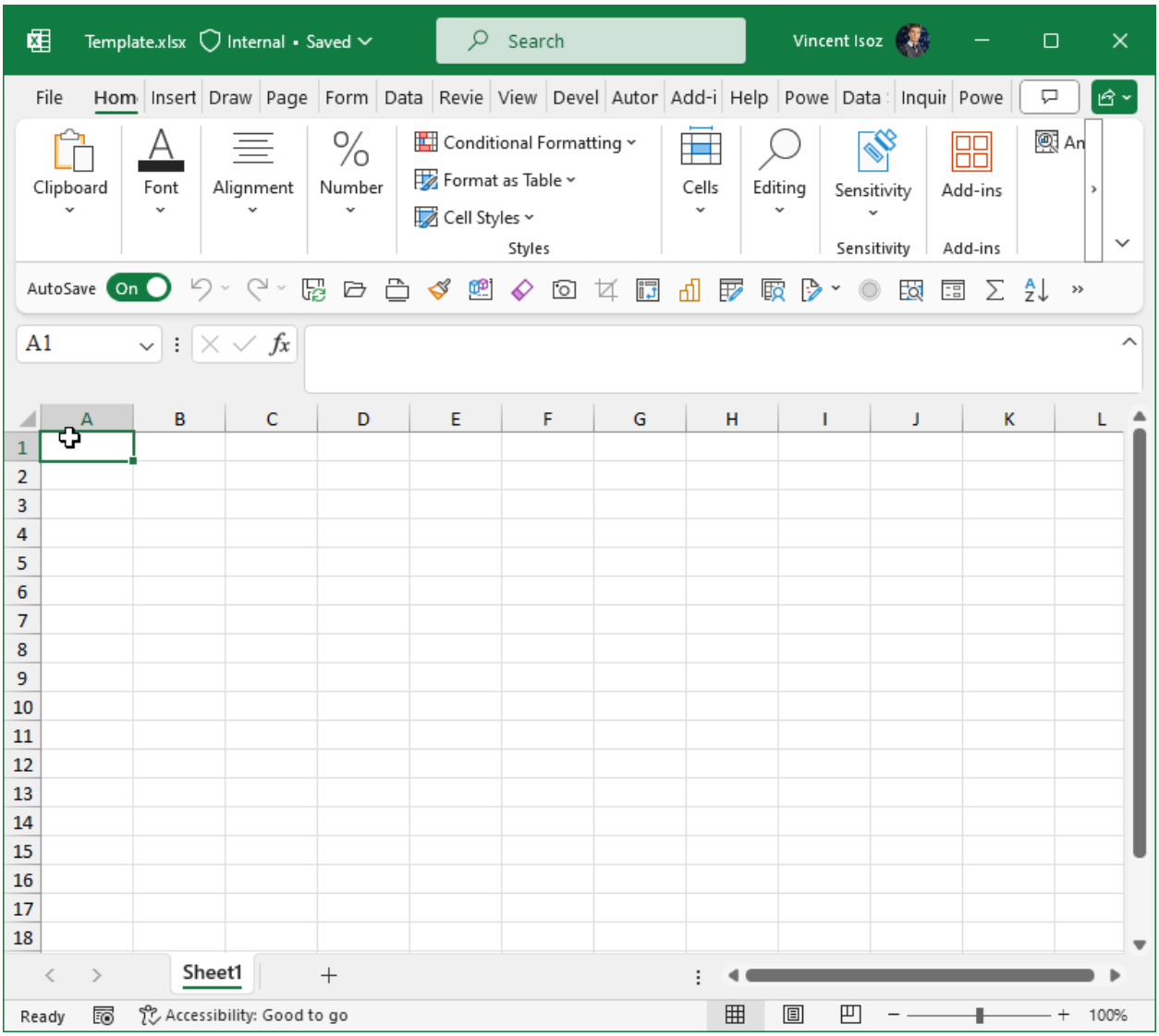
Qui contiennent respectivement:



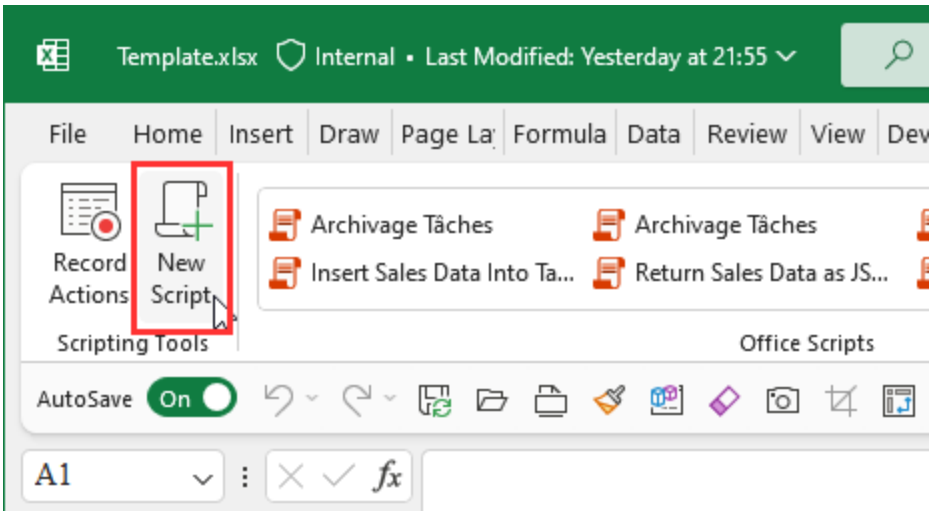
Et:



Ainsi que:



Maintenant, nous créons un script:



Pour les fichiers CSV avec des données séparées par des "," le script est:

Public
Internal

Code Editor

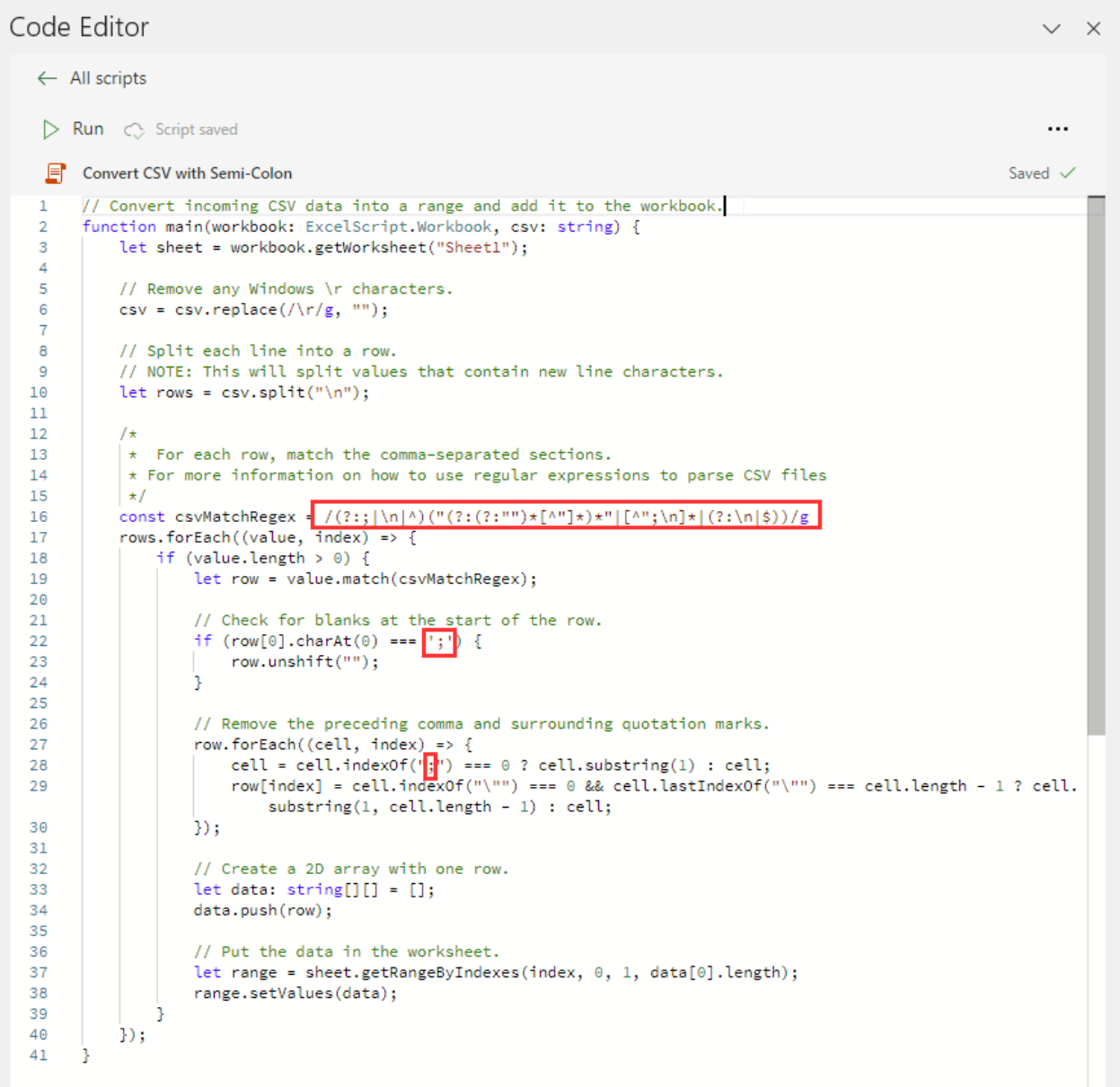
All scripts

Run Script saved

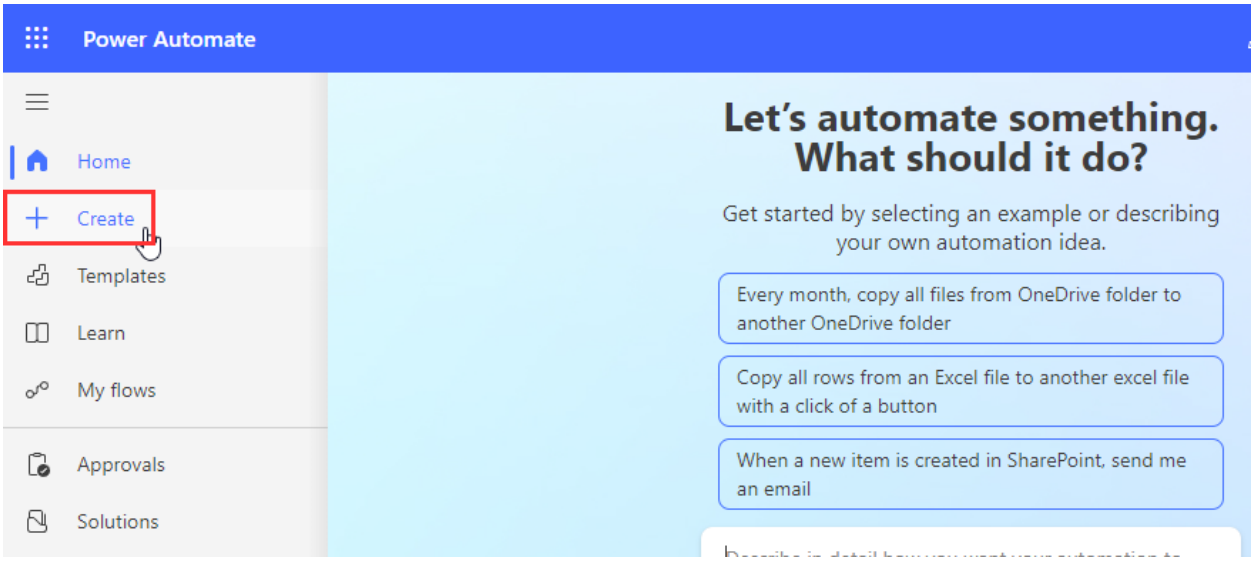
Convert CSV with Comma Saved

```
// Convert incoming CSV data into a range and add it to the workbook.
function main(workbook: ExcelScript.Workbook, csv: string) {
    let sheet = workbook.getWorksheet("Sheet1");

    // Remove any Windows \r characters.
    csv = csv.replace(/\r/g, "");

    // Split each line into a row.
    // NOTE: This will split values that contain new line characters.
    let rows = csv.split("\n");

    /*
     *  For each row, match the comma-separated sections.
     * For more information on how to use regular expressions to parse CSV files
     */
    const csvMatchRegex = /(?:,|\n|^)("(?:(?:"")*[^"]*)*"|[^",\n]*|(?:\n|$))/g
    rows.forEach((value, index) => {
        if (value.length > 0) {
            let row = value.match(csvMatchRegex);

            // Check for blanks at the start of the row.
            if (row[0].charAt(0) === ',') {
                row.unshift("");
            }

            // Remove the preceding comma and surrounding quotation marks.
            row.forEach((cell, index) => {
                cell = cell.indexOf(",") === 0 ? cell.substring(1) : cell;
                row[index] = cell.indexOf("\"") === 0 && cell.lastIndexOf("\"") === cell.length - 1 ? cell.
                    substring(1, cell.length - 1) : cell;
            });

            // Create a 2D array with one row.
            let data: string[][] = [];
            data.push(row);

            // Put the data in the worksheet.
            let range = sheet.getRangeByIndexes(index, 0, 1, data[0].length);
            range.setValues(data);
        }
    });
}
```

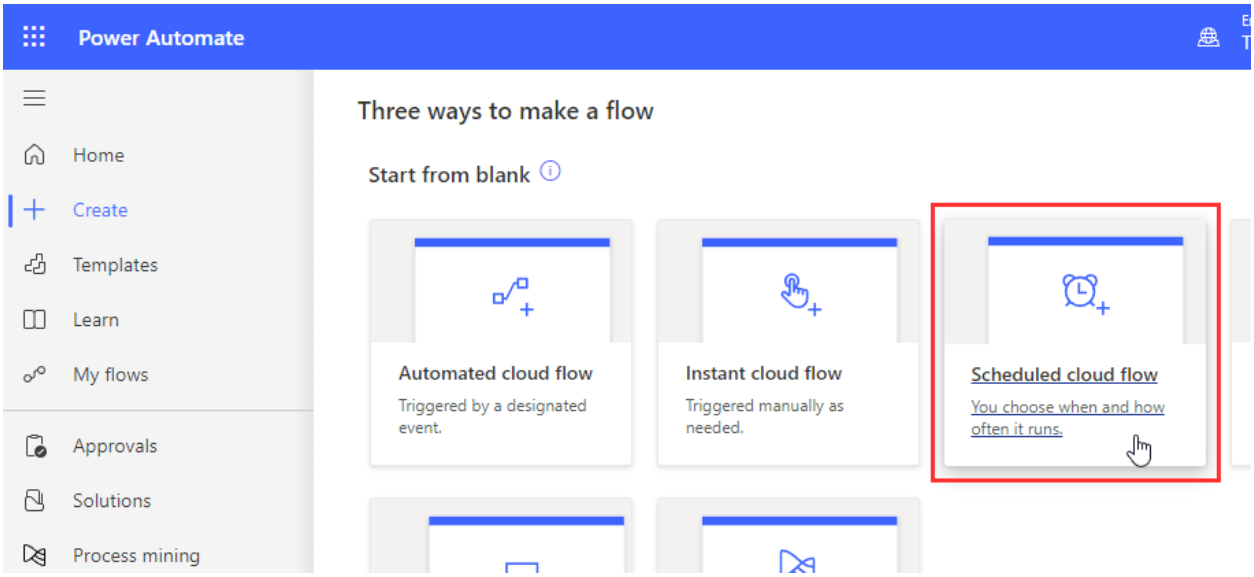
Et pour les fichiers CSV dont les données sont séparées par des ";" nous avons (on a mis en rouge ce qui change):



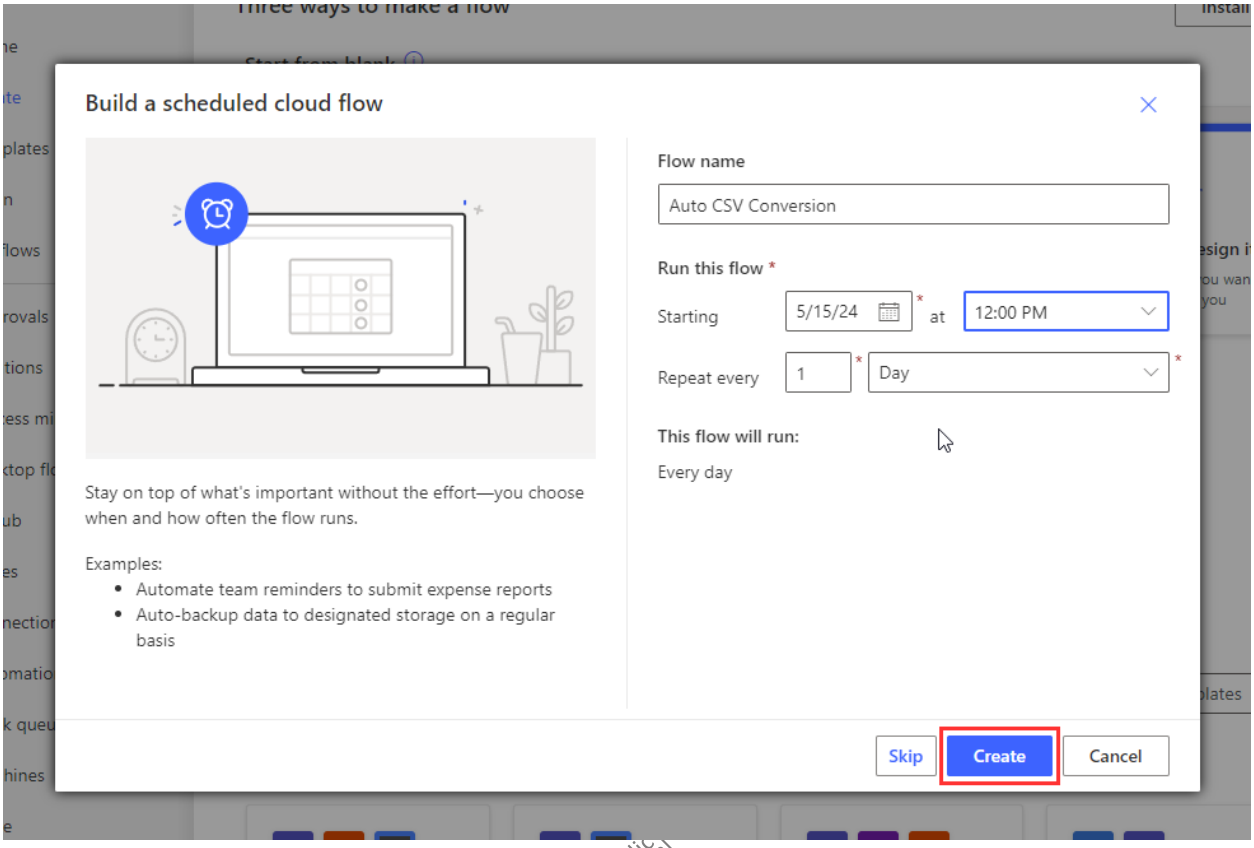
Maintenant, nous ouvrons Power Automate et cliquons sur **Create**:



Prenons **Scheduled cloud flow**:

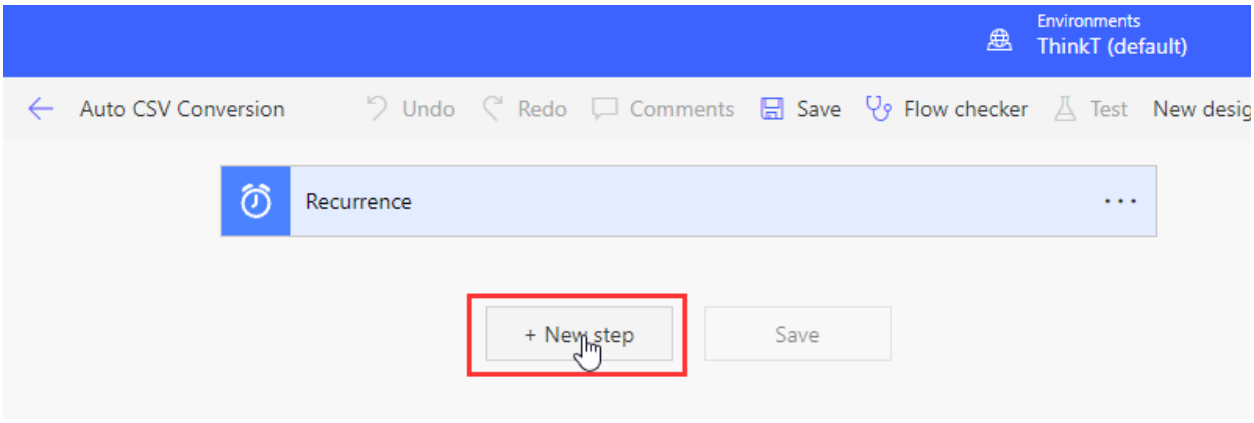


Prenons typiquement les options suivantes:

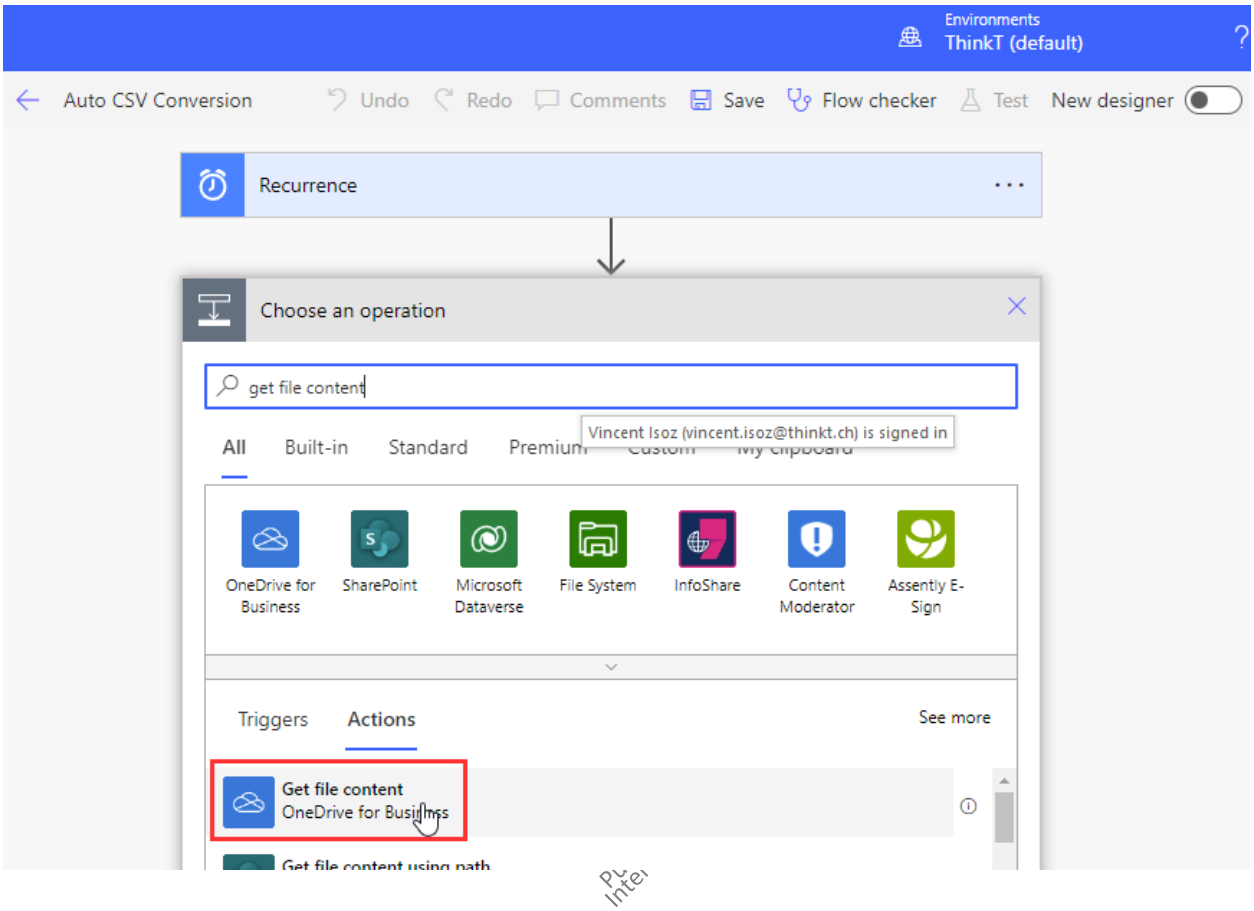


Public External

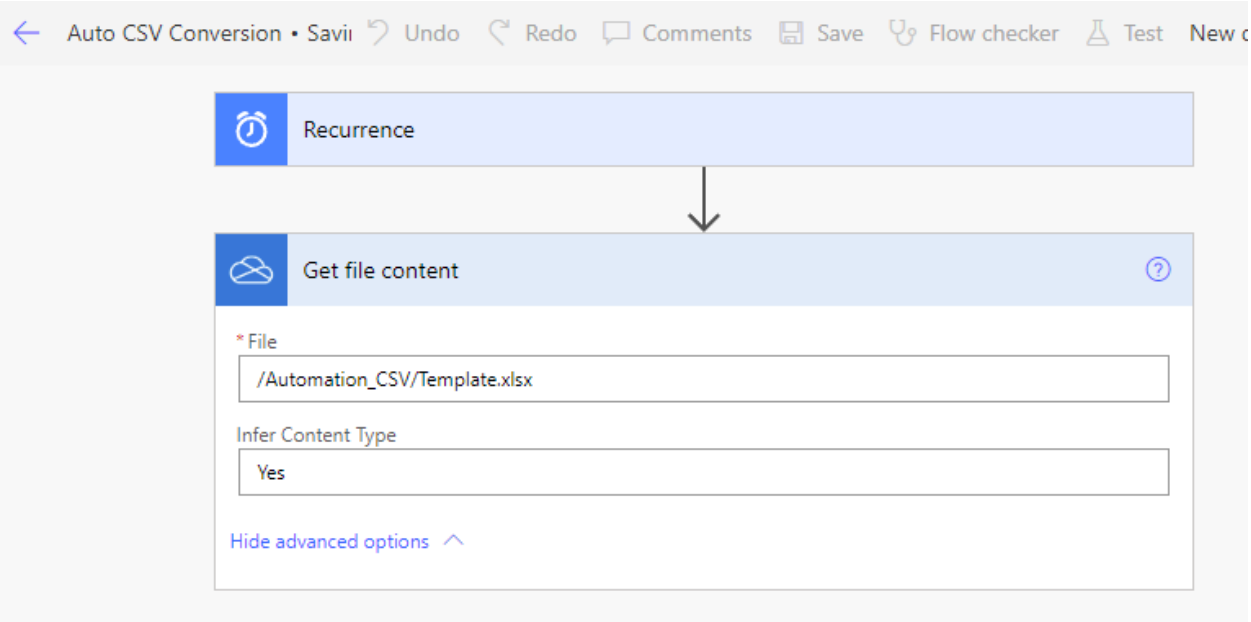
Et cliquons sur **Create**. Ensuite ajoutons une étape:



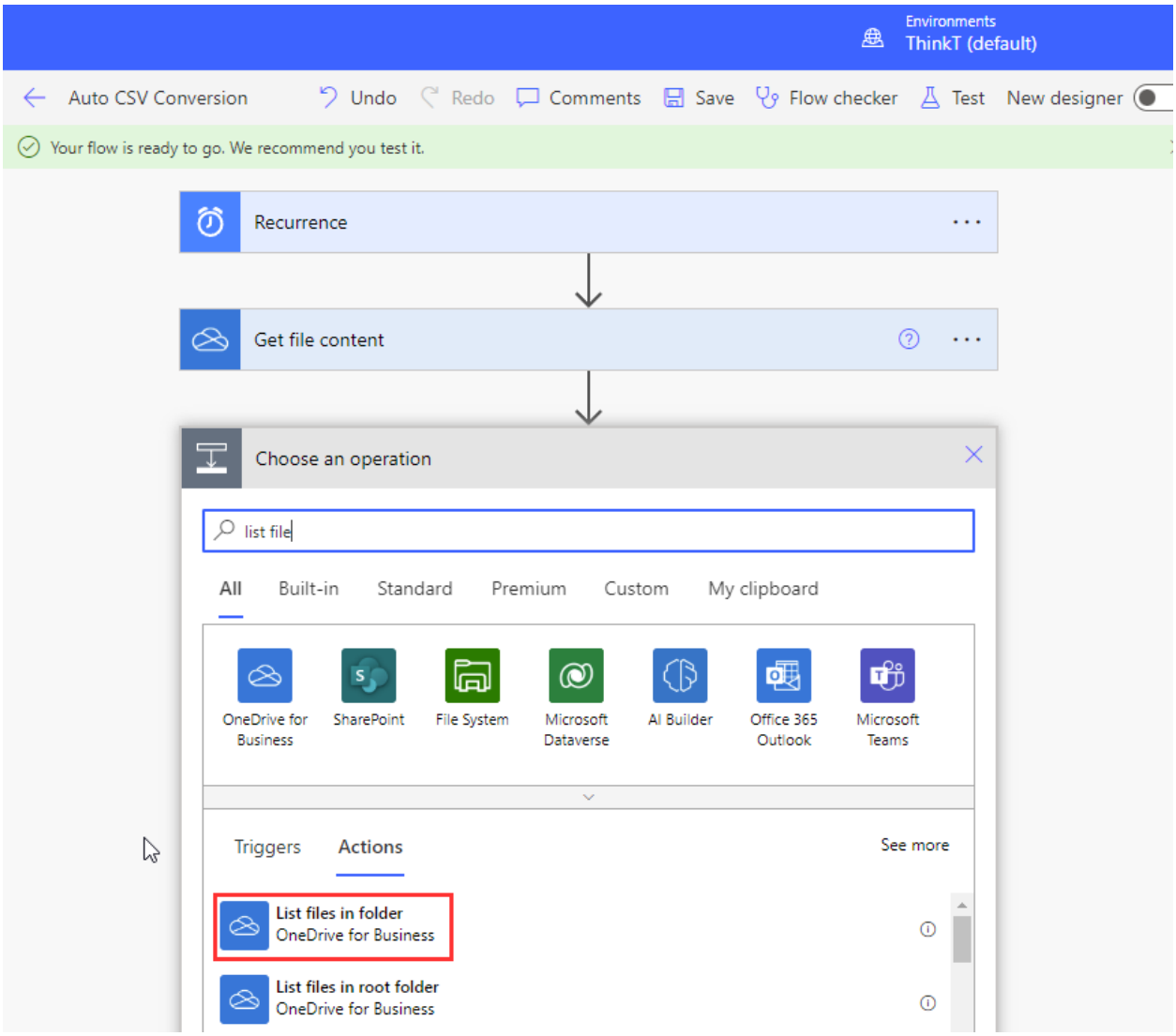
Nous prenons l'action *Get file content*:



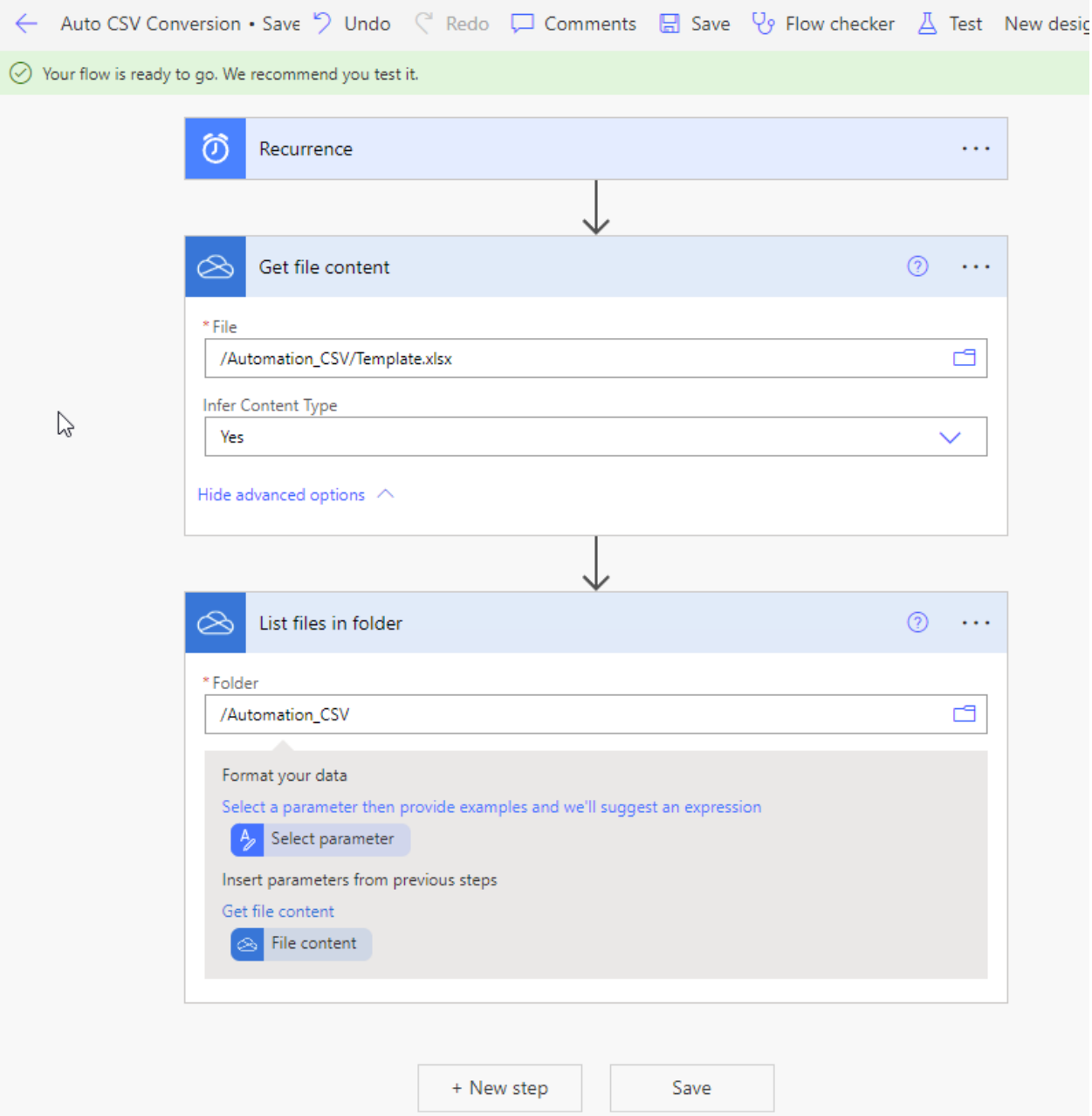
Et nous prenons:



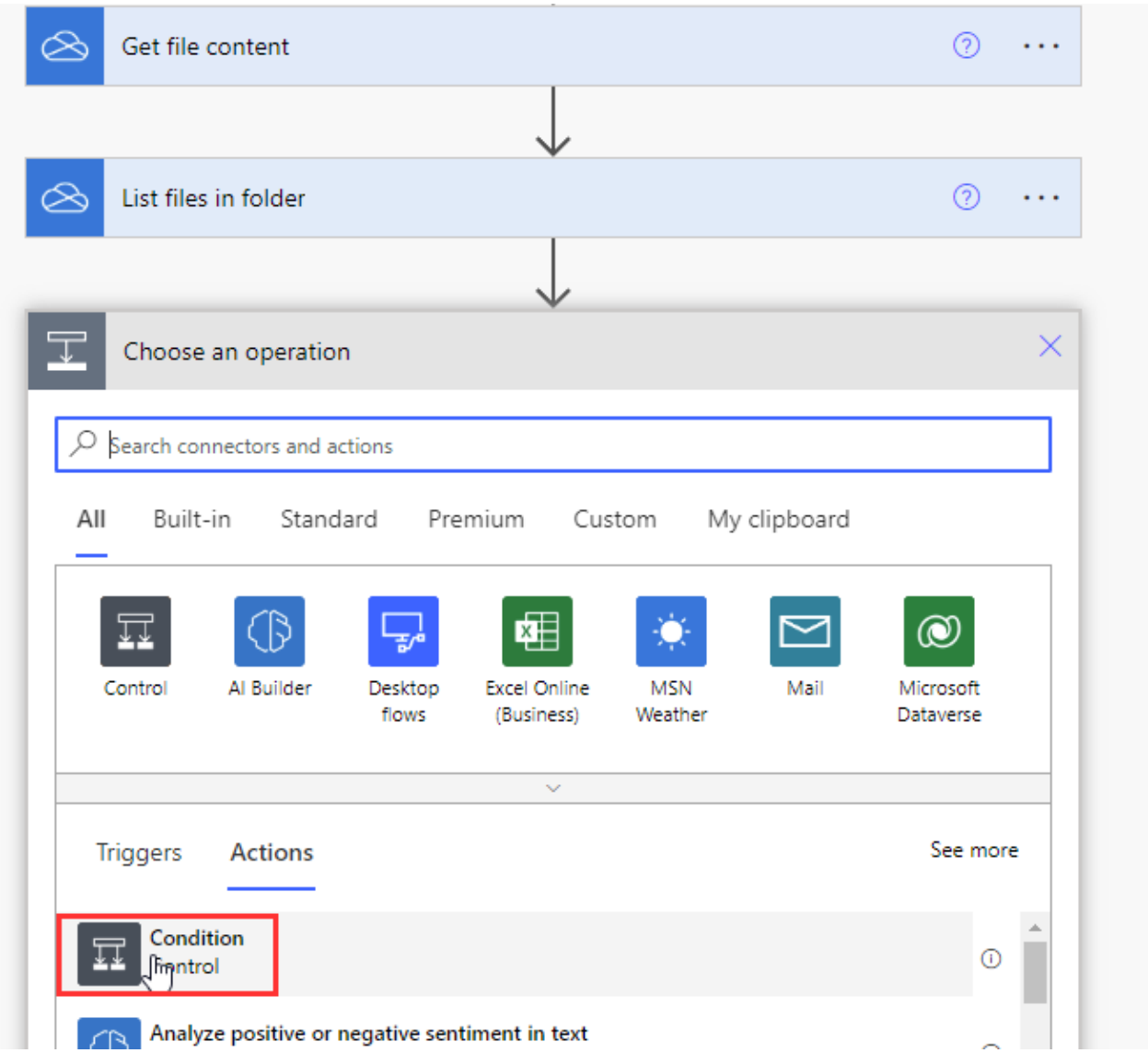
Nous prenons maintenant l'action **List files in folder**:



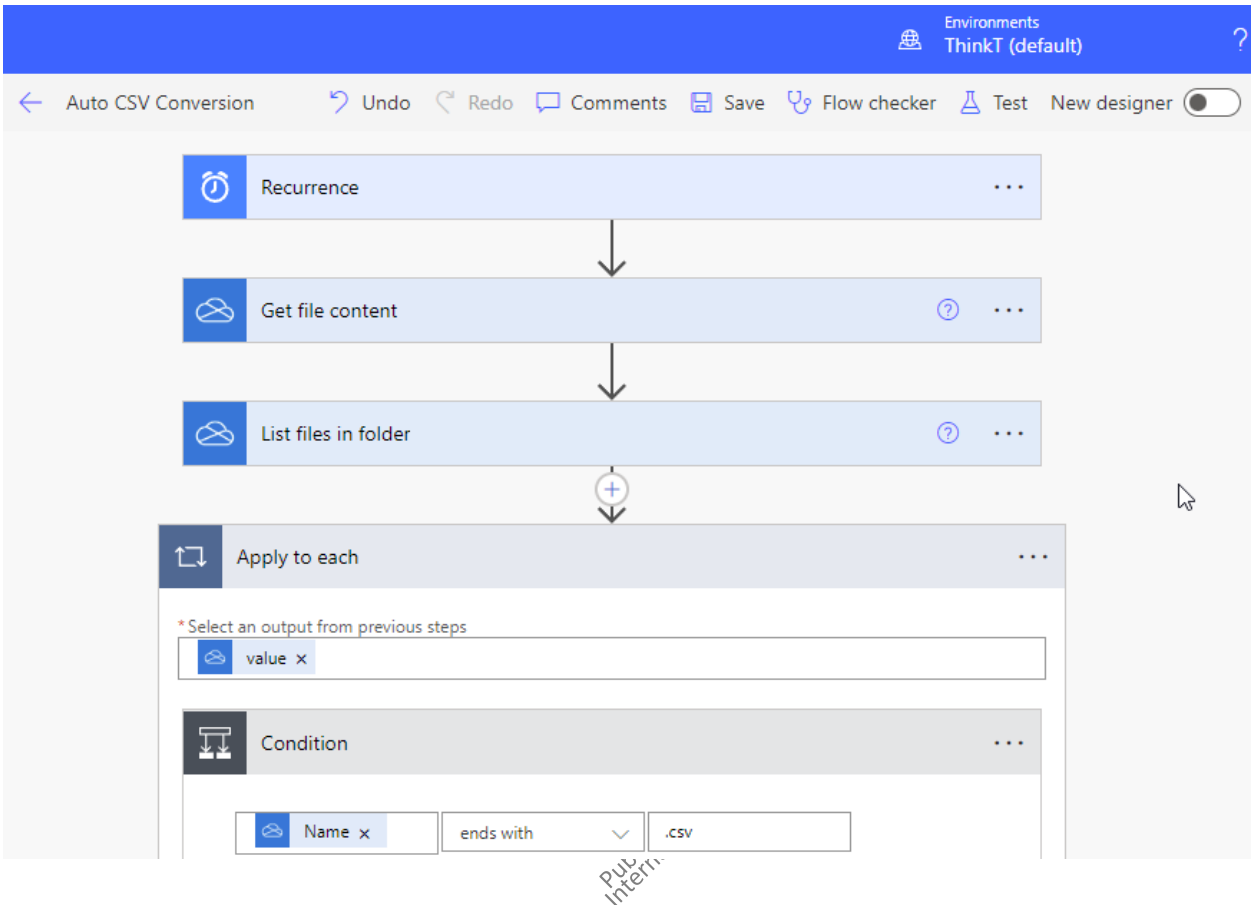
Et nous prenons le dossier qui contient les fichiers *.csv:



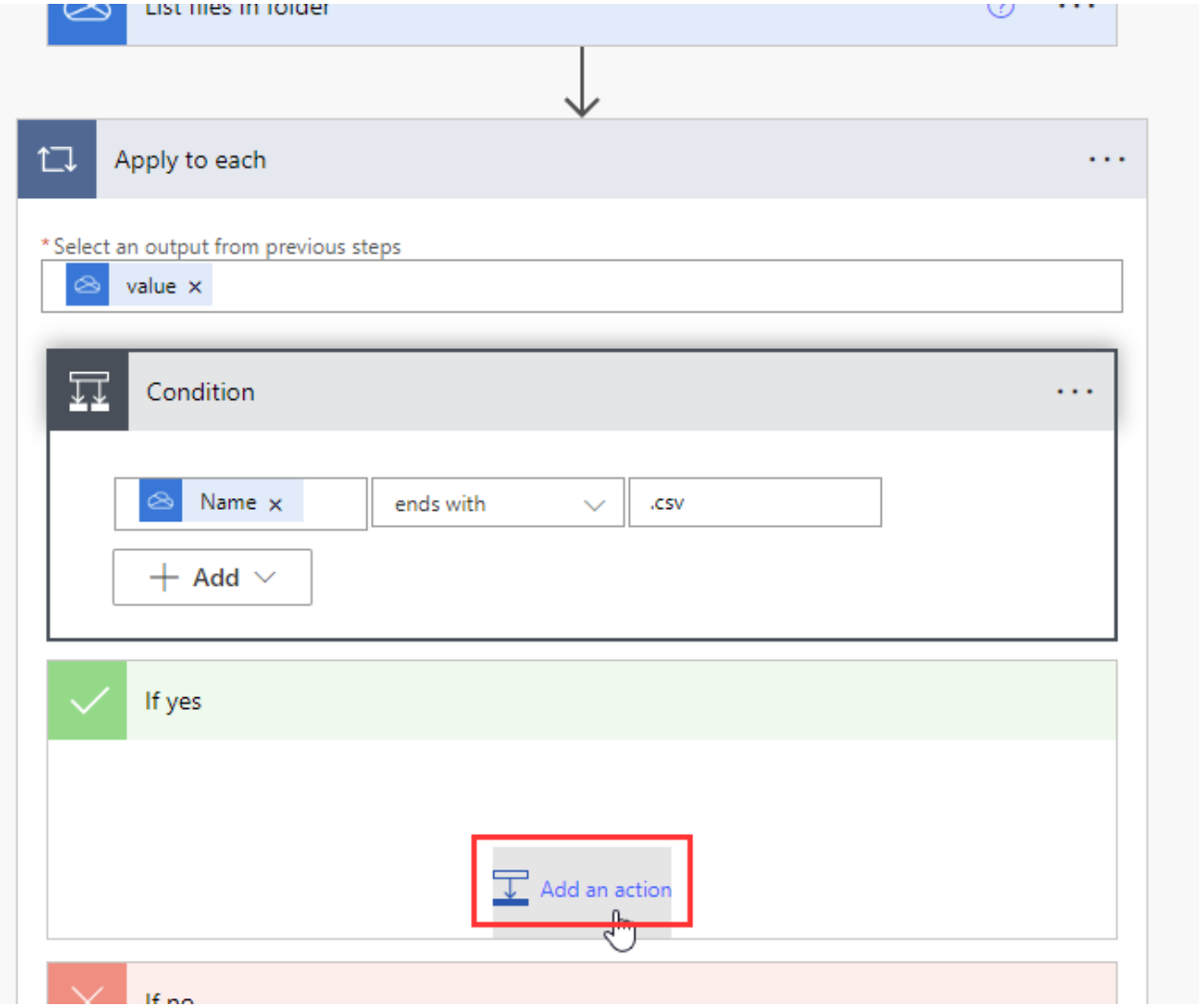
Ajoutez une condition pour que le flux fonctionne uniquement sur les fichiers .csv. Ajoutez l'action de contrôle *Condition*:



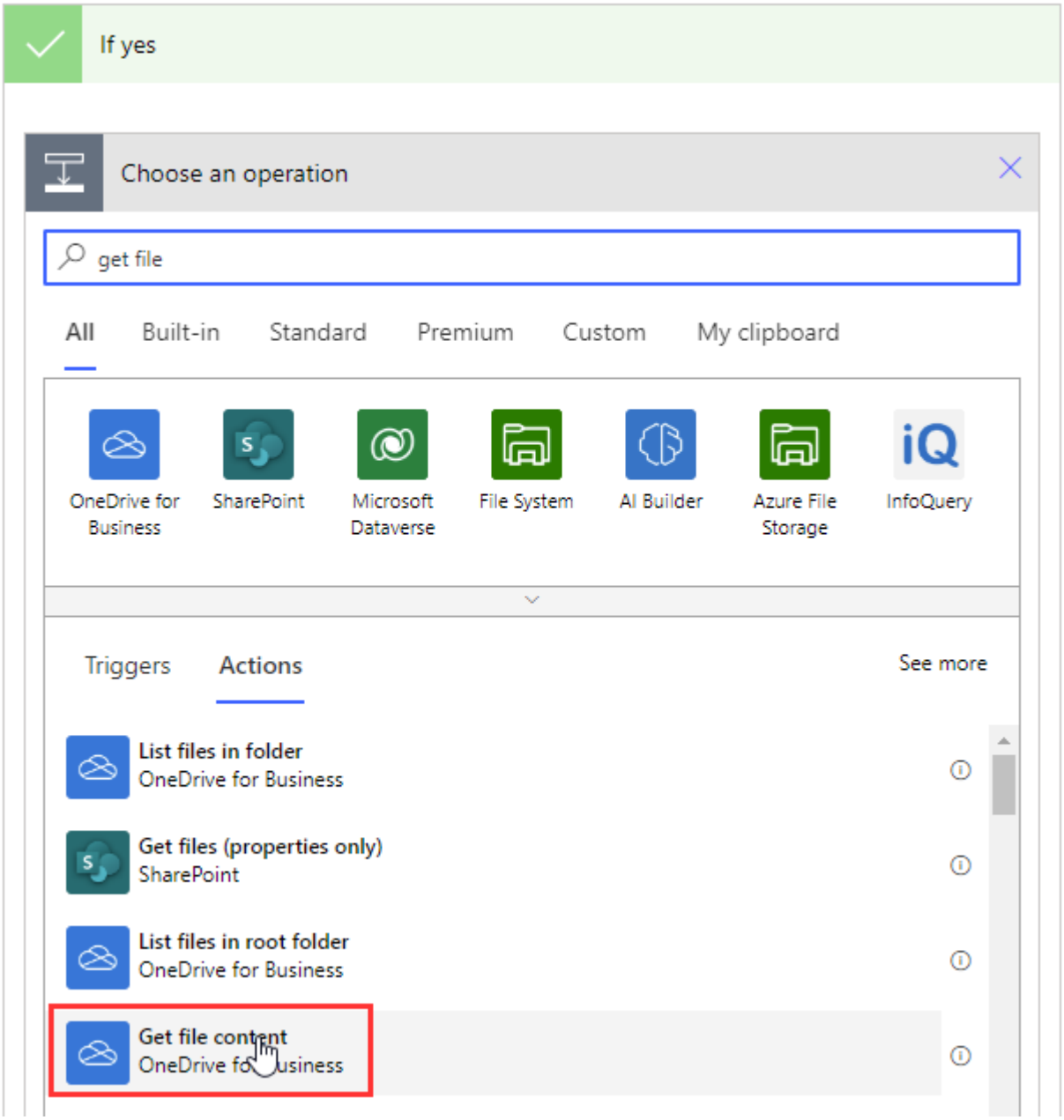
Utilisez les valeurs suivantes pour la condition (le *Apply to each*) se créé automatiquement:



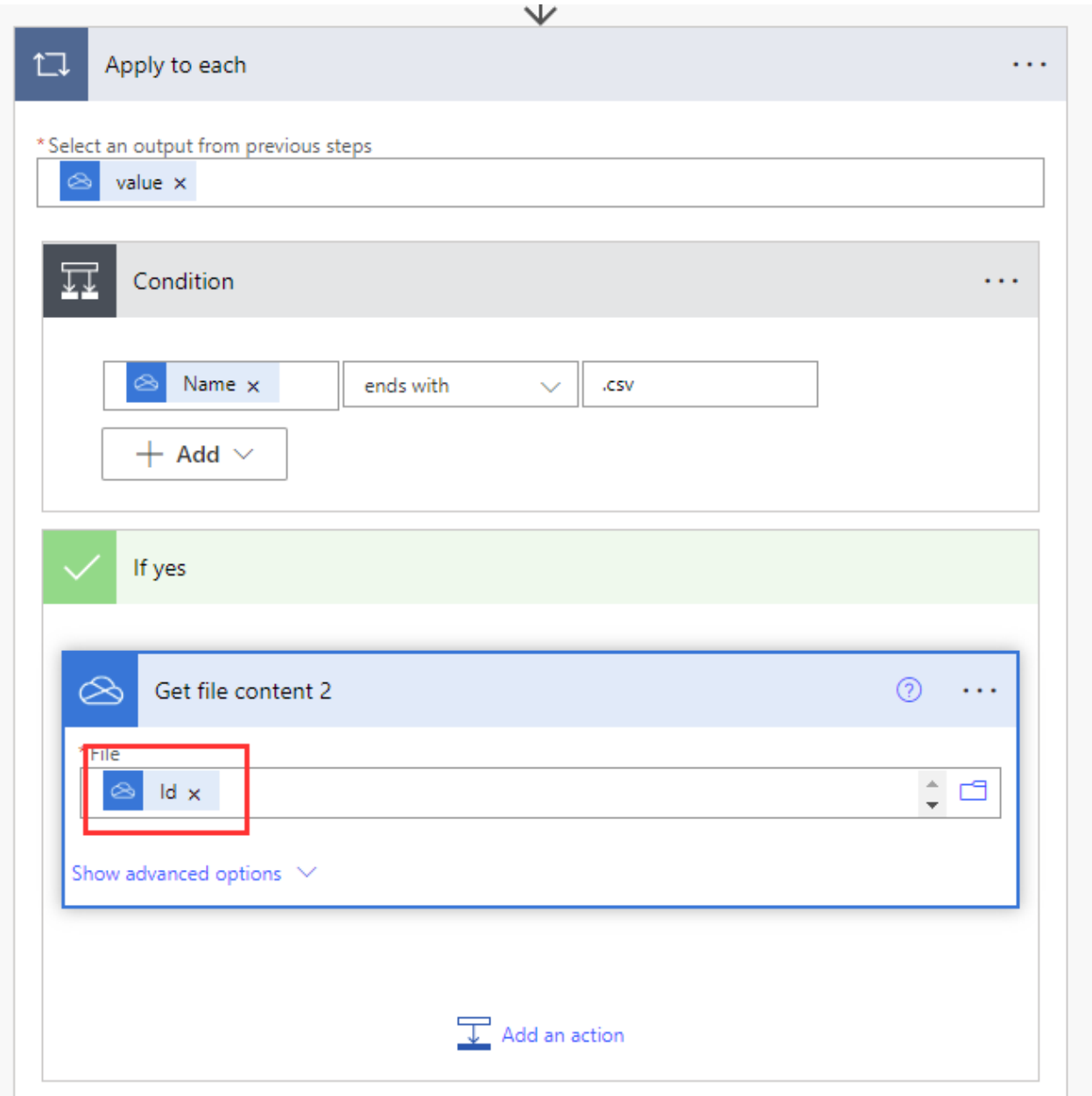
Ensuite, nous allons dans le *Add an action* de la partie *If Yes*:



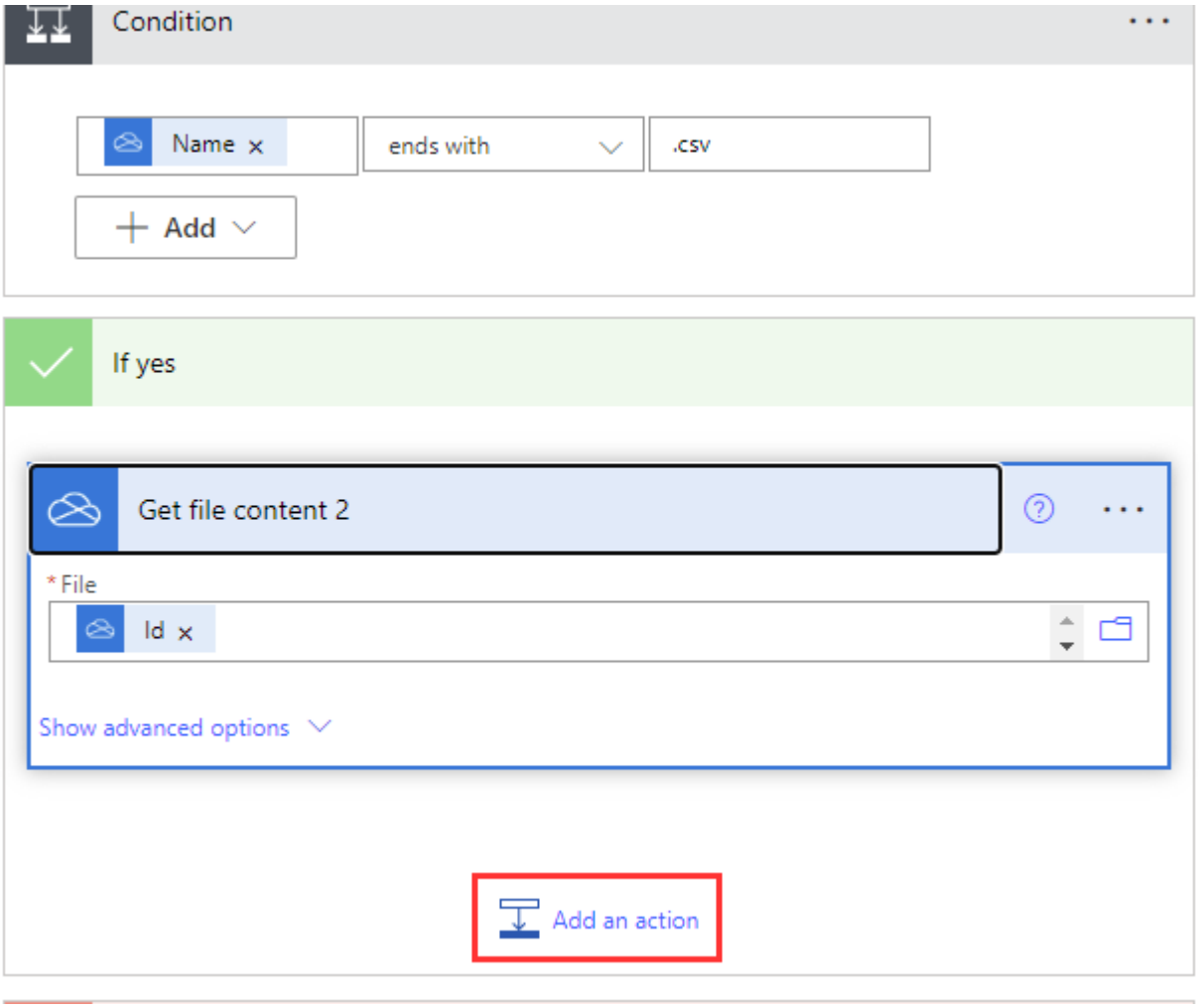
Et nous prenons l'action *Get file content*:



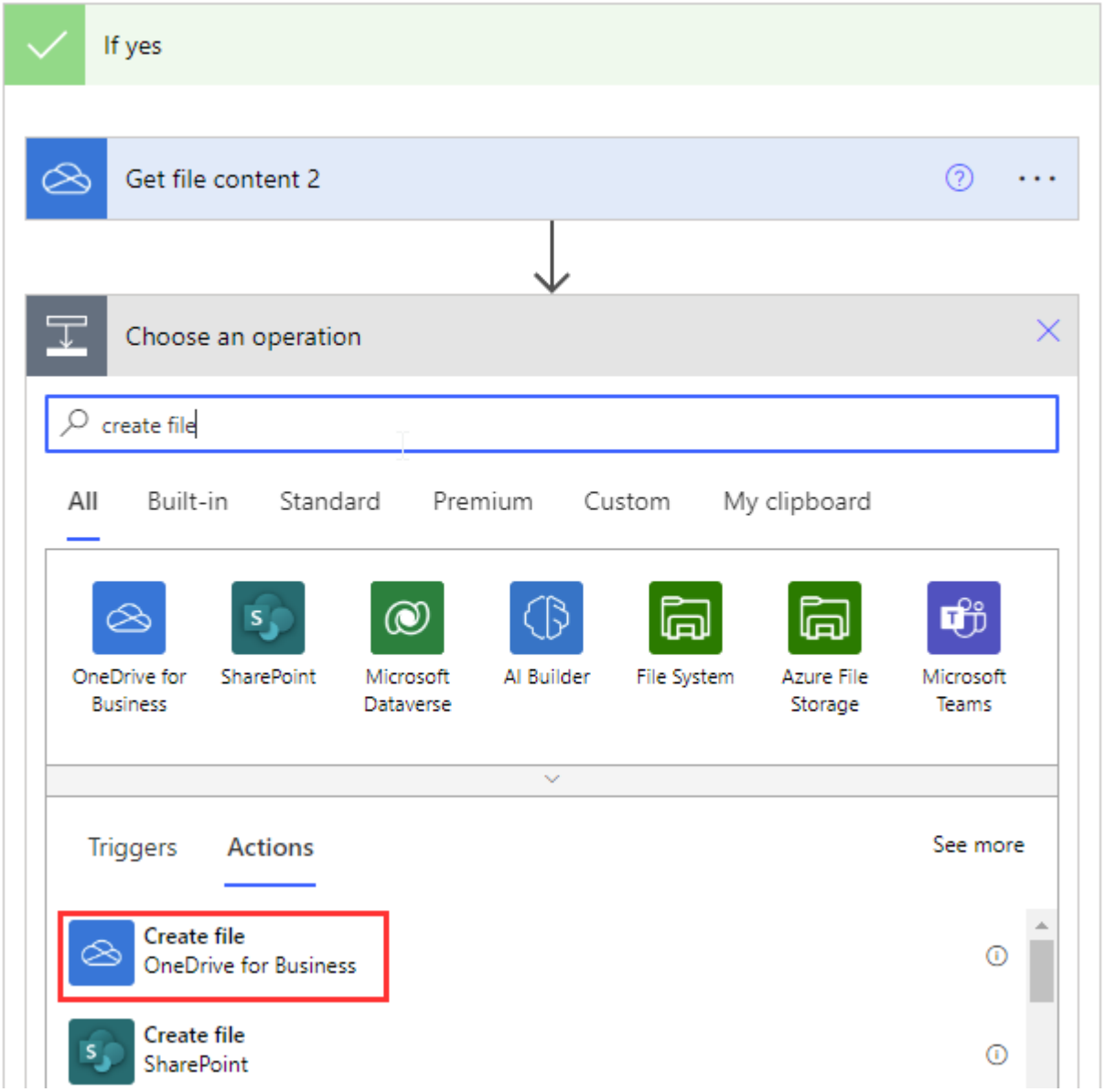
Et nous prenons:



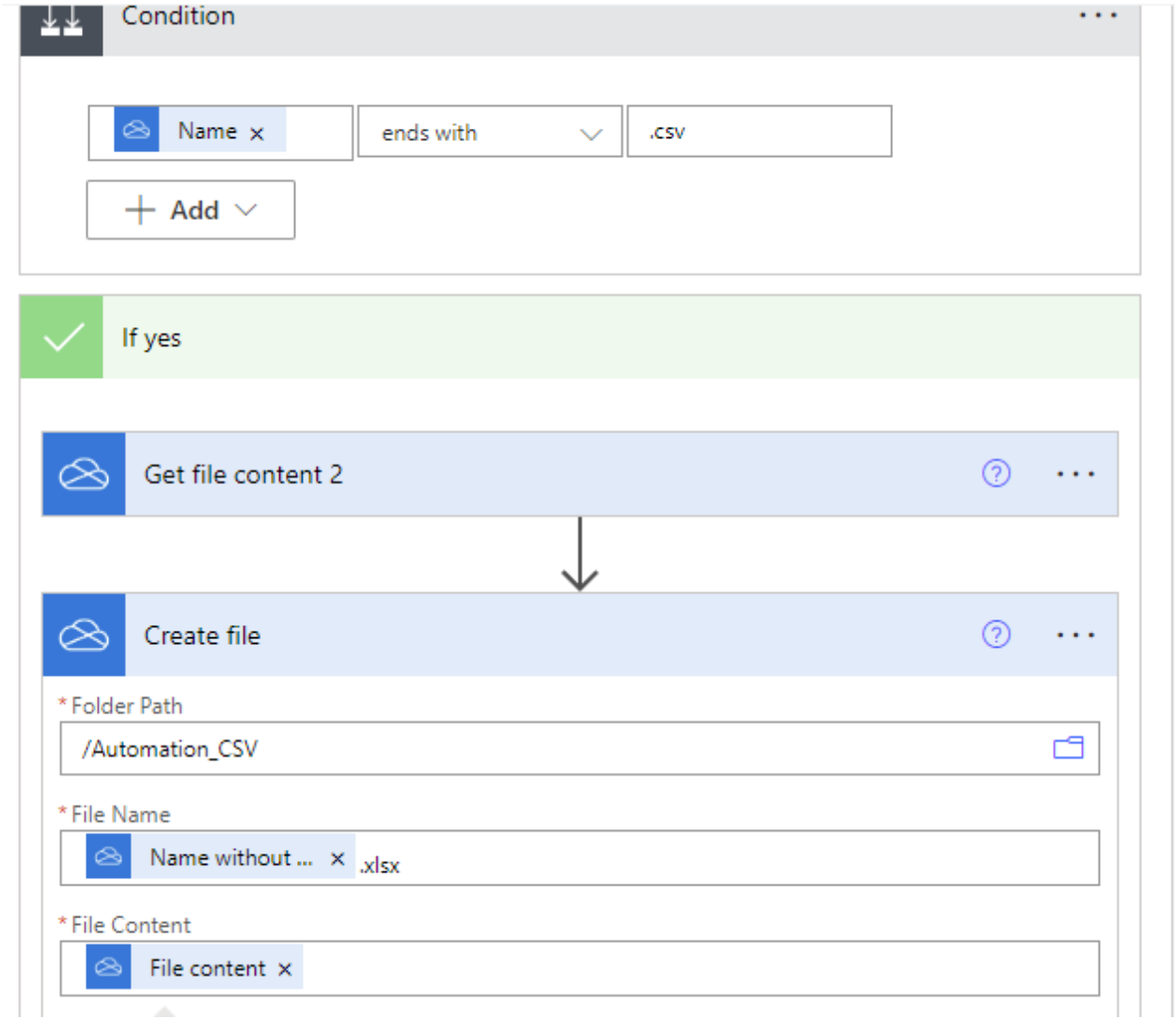
Nous cliquons à nouveau sur *Add an action*:



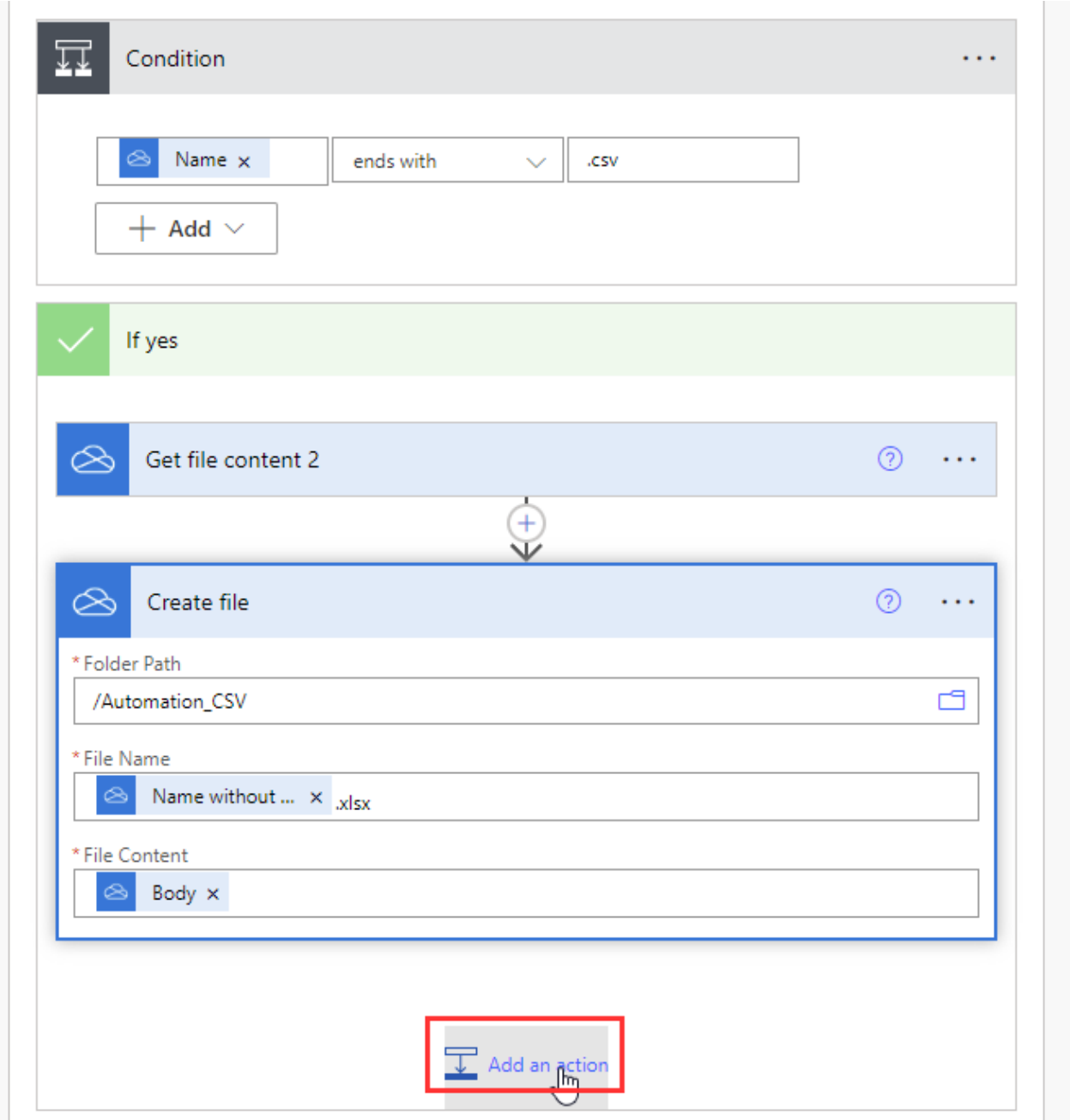
Et nous prenons l'action *Create file*:



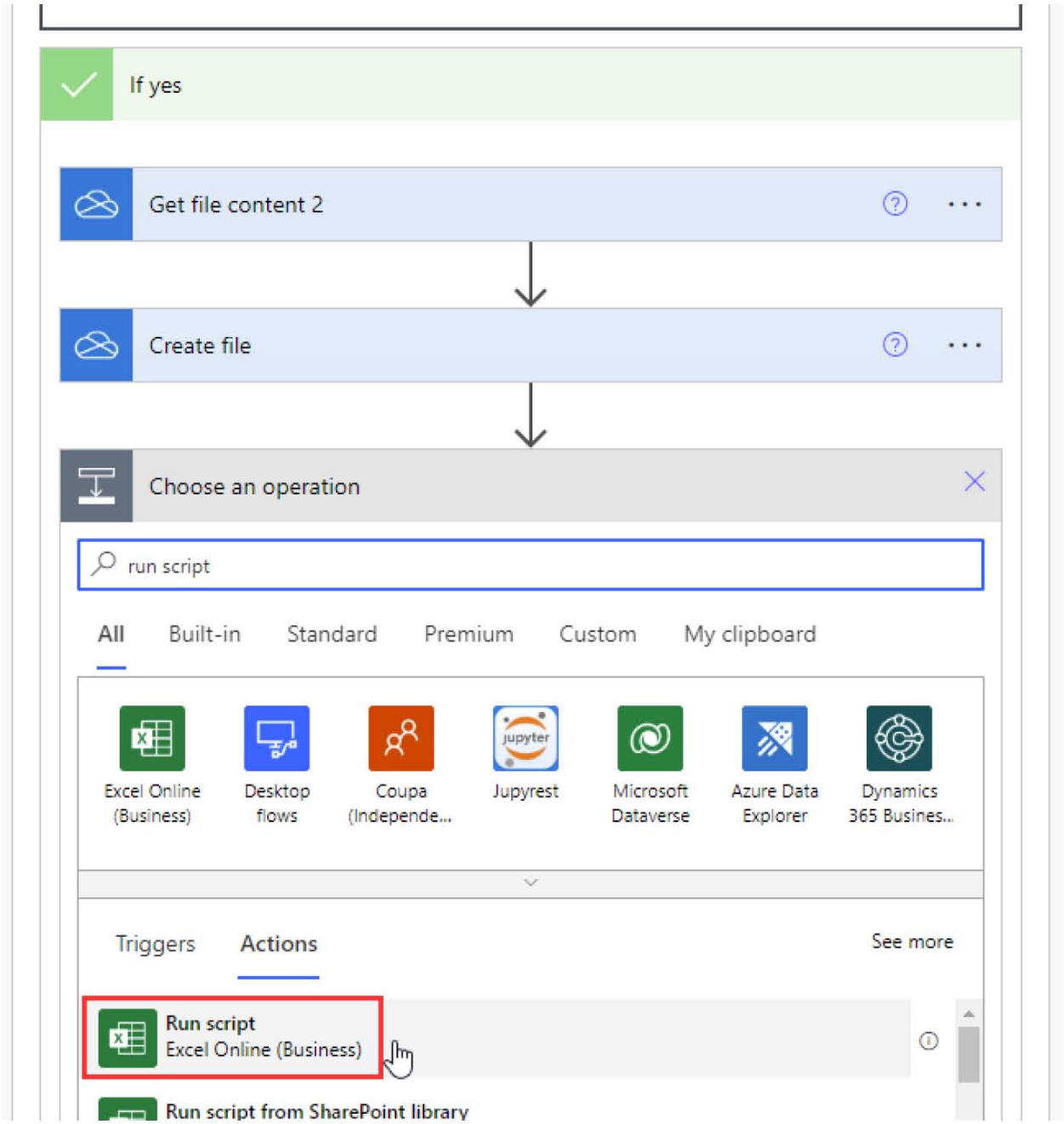
Et nous prenons:



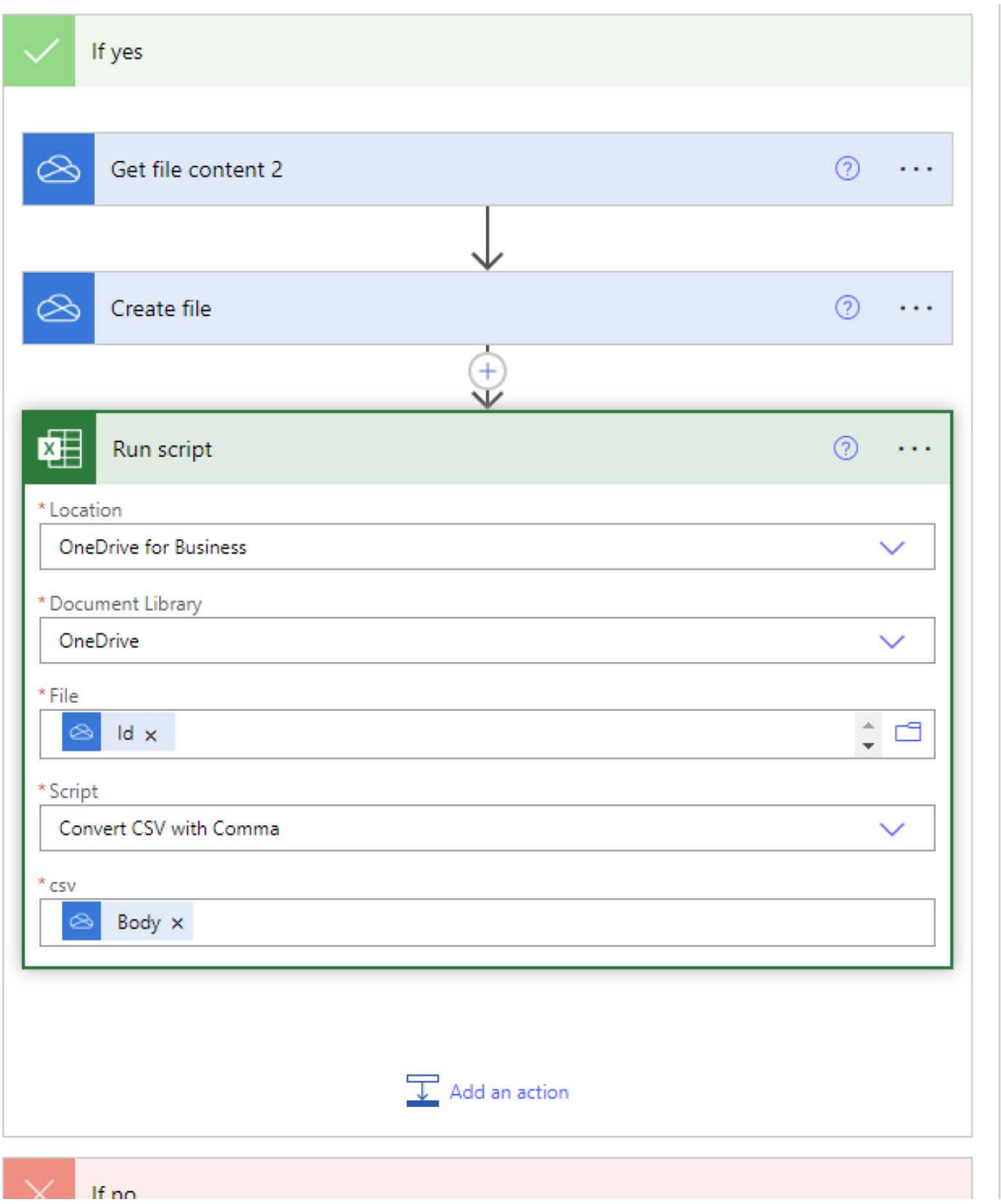
Et nous ajoutons encore une action:



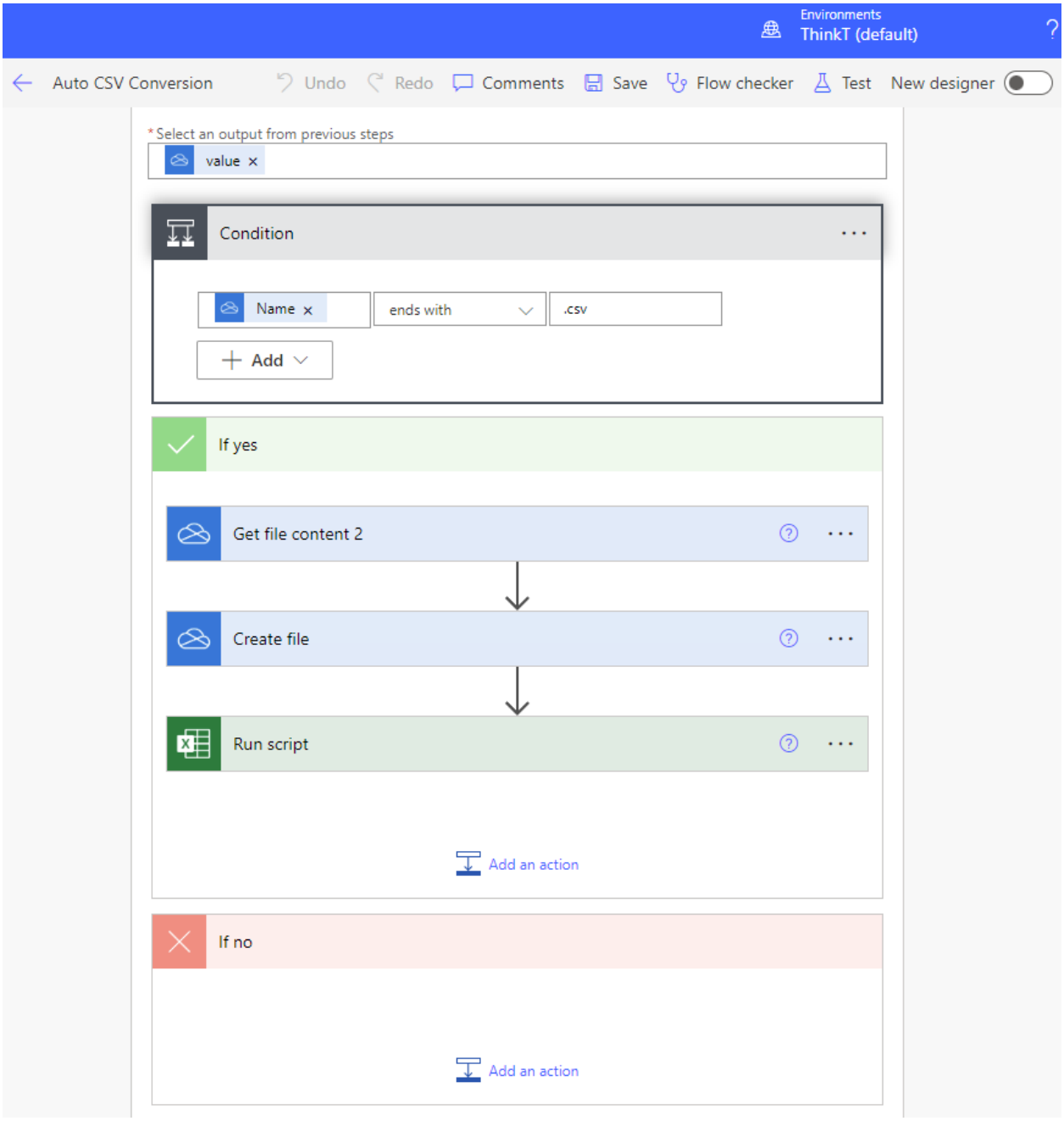
Et nous prenons l'action *Run script*:


If yes
Get file content 2
Create file
Choose an operation
run script
All
Built-in
Standard
Premium
Custom
My clipboard
Excel Online (Business)
Desktop flows
Coupa (Independe...
Jupyrest
Microsoft Dataverse
Azure Data Explorer
Dynamics 365 Busines...
Triggers
Actions
See more
Run script
Excel Online (Business)
Run script from SharePoint library

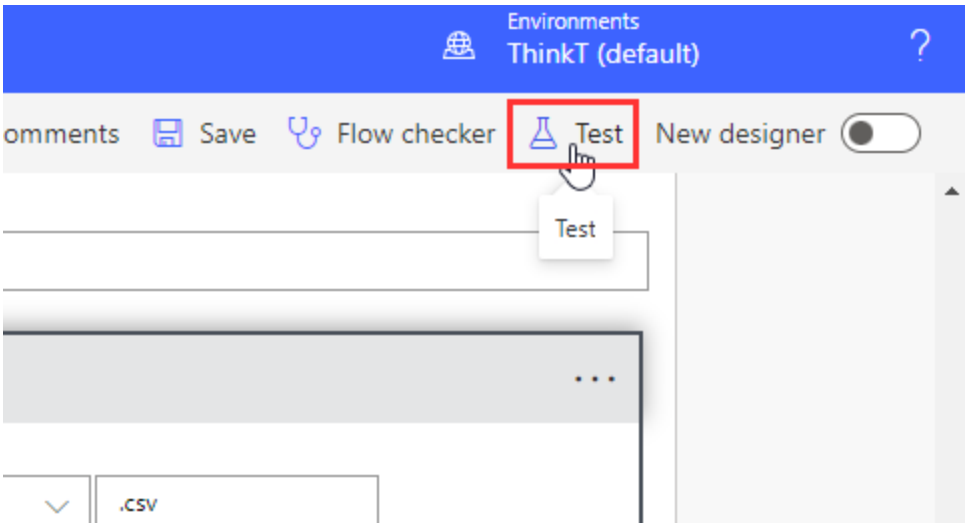
Avec:



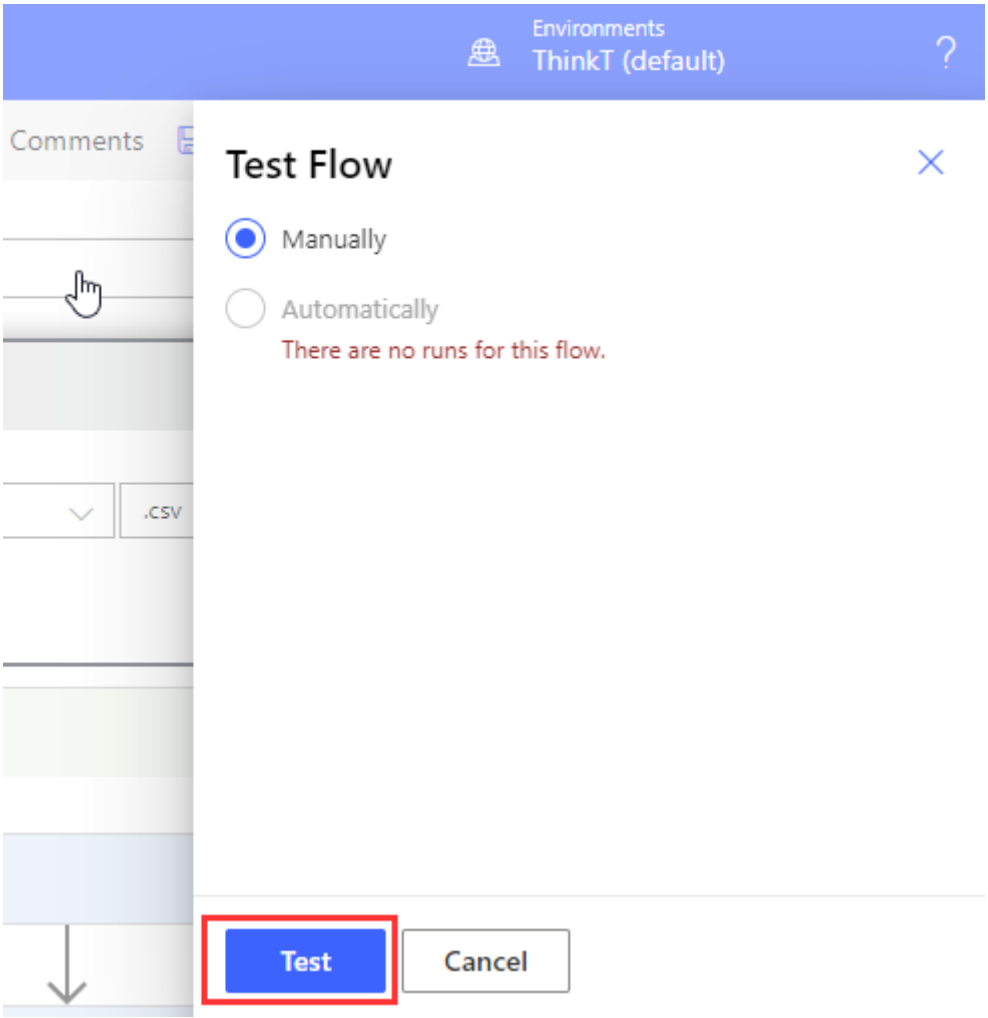
Sauvegarder le flux qui devrait ressembler au final à:



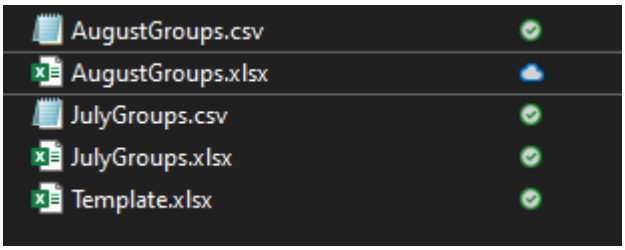
Nous pouvons maintenant cliquer sur le bouton **Test**:



Et nous cliquons sur **Test**:



Ce qui aura pour effet de créer un fichier XLSX par fichier CSV:



Public
Internal

15 Conclusion

Nous avons vu au travers de ce PDF les bases de l'Office Script. Est-ce toutefois la fin de VBA ?

Les scripts Office sont-ils destinés à remplacer VBA ? Je pense que oui, mais pas de si tôt (de mon vivant du moins mais peut-être je me trompe!).

De nombreuses entreprises utilisent la version Desktop d'Excel, un domaine dans lequel les scripts Office sont peux connus car disponibles que depuis 2023. Il faudra encore quelques décennies avant que toutes les organisation passent pas à Office 365.

Beaucoup de temps et d'argent ont été investis dans des solutions VBA complexes pour automatiser de nombreux processus métier. Ces processus ne peuvent actuellement pas être mis à niveau vers les scripts Office en raison d'une incompatibilité ou d'un manque de fonctionnalités.

Il est probable que VBA existera encore pendant une dizaine d’années et il sera très vraisemblablement pas remplacé par Office Script dans Microsoft Excel mais plutôt par Power BI et son association R et Python.

Public
Internal

16 Table des figures

Figure 1 Évolution des langages dans Excel..... 8

Figure 2 Pong sur ATARI en circuit logique..... 20

17 Index

No index entries found.

Public
Internal